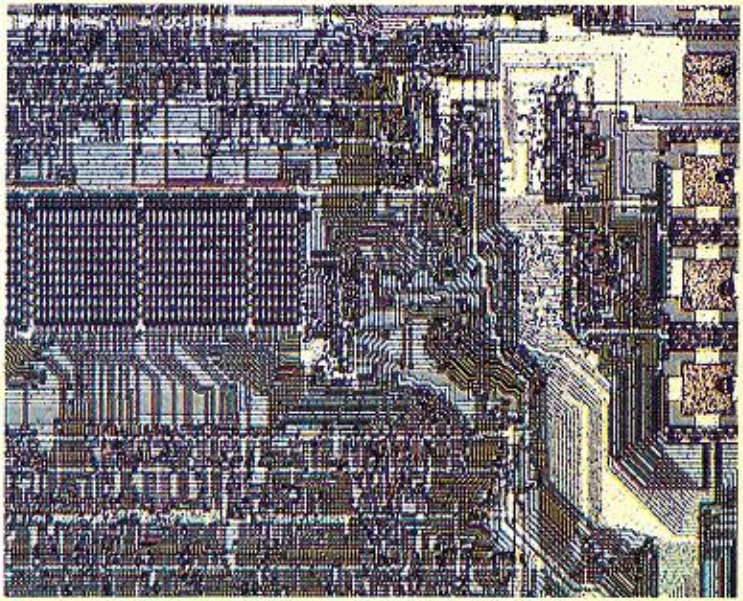
MO5

MANUEL DE L'ASSEMBLEUR 6809 DU MO5

Michel Weissgerber



TO TEK International

cedic
nathan

Manuel de l'assembleur 6809 sur MO5

Michel Weissgerber

TO TEK International

cedic nathan

Illustrations : Annicka Boyriven

Ce volume porte la référence
ISBN 2-7124-0570-6

Toute reproduction, même partielle, de cet ouvrage est interdite. Une copie ou reproduction par quelque procédé que ce soit, photographie, photocopie, microfilm, bande magnétique, disque ou autre, constitue une contrefaçon passible des peines prévues par la loi du 11 mars 1957 sur la protection des droits d'auteur.

© CEDIC 1984

CEDIC, 32, boulevard Saint-Germain, 75005 - PARIS

SOMMAIRE

Avant-propos 7

1 - Organisation de la cartouche Assembleur 9

1.1 - Architecture 10

1.2 - Menu 11

1.3 - Descripteur de fichier 12
Nom de fichier 12

2 - L'éditeur 13
Accès à l'éditeur 15
Écriture d'un programme source 16
Les programmes d'essai 21
Exemple : « EQUATE » 22
Sauvegarde d'« EQUATE » 23
Exemple : « MOIRAGE » 24
Sauvegarde de « MOIRAGE » 25

2.1 - Commandes d'édition 26
Déplacement du curseur (vers le haut) 26
(vers le bas) 27
(vers la gauche) 28
(vers la droite) 28
Espace 29
Entrée 29
Insertion d'un caractère 30
Effacement d'un caractère 31
Descente d'une page 32
Remontée d'une page 33
Effacement d'une ligne 34
Positionnement du curseur (en fin de ligne) 35
(au début de la ligne) 35
Caractères spéciaux 36
Affichage d'un espace en vidéo inversée 36

Commande de changement d'état
(mode édition-commande éditeur) 37

2.2 - Commandes éditeur 38
TOP (début du programme source résidant dans l'éditeur) 38
BOTTOM (fin du programme source résidant dans l'éditeur) 39
OFF (suppression de la tabulation automatique) 39
ZONE (sélectionne une zone mémoire dans l'éditeur) 40
COPY (sauvegarde une partie du programme source) 41
INSERT (insertion dans le programme source) 42
DELETE (effacement d'une ligne ou de la zone) 43
FIND (recherche d'une chaîne de caractères) 44
REPLACE (remplacement d'une chaîne de caractères) 46
PRINT (impression sur l'imprimante) 49
NEW (effacement) 50
EXIT (passage sous contrôle du moniteur) 51
QUIT (retour sous contrôle de menu) 52

2.3 - Commandes de gestion de fichiers sous éditeur 53
LOAD (chargement dans l'éditeur d'un programme source) 53
SAVE (sauvegarde le programme source résidant) 55
MERGE (fusionne un fichier avec le programme source) 56
VERIFY (vérification du contenu de l'éditeur) 57
END OF FILE (positionnement du LEP) 58

3 - Assemblage 59

ASSEMBLAGE (assemblage du programme source) 61

3.1 - Options d'assemblage 64
/WE (arrête l'assemblage si une erreur est détectée) 64
/NL (supprime l'affichage des listings) 65
/LP (commutation de l'imprimante) 66
/NO (suppression du programme objet en mémoire) 66
/NS (suppression de la table des symboles) 67
/SS (édition sur des lignes séparées) 67

3.2 - Directives d'assemblage 68
EQU (affectation d'une valeur à un symbole) 68
SET (affectation temporaire d'une valeur à un symbole) 69
RMB (réservation d'octets en mémoire) 70
ORG (initialisation du compteur de programme) 71
FCB (définition d'une constante d'un octet) 73
FDB (définition d'une constante de deux octets) 74

FCC (définition d'une constante chaîne de caractères) 75
SETDP (positionnement de la page directe « Ø ») 76
PAGE (saut de page) 78
TITLE (définition d'un titre) 78
INCLUD (inclusion d'un programme source) 79
END (fin du programme source) 80
CALL (appel d'un sous-programme moniteur) 81
GOTO (saut à un programme moniteur) 82
STOP (retour en état « attente de commande moniteur ») 83
Où est situé le programme objet ?
Quelle est la fin du programme source ? 84

4 - Le moniteur 88
Accès au moniteur 88

4.1 - Commandes moniteur 90
INPUT (entrée) 91
(ADRESSE) / (examen du contenu d'un emplacement mémoire) ... 91
MODE NUMÉRIQUE 93
OUTPUT (sortie) 94
MODE ASCII 95
MODE MNÉMONIQUE 96
IMPRESSION SYMBOLIQUE 97
IMPRESSION NUMERIQUE 98
IMPRESSION MI-SYMBOLIQUE MI-NUMERIQUE 99
INDIRECTION (introduit un niveau d'indirection dans l'examen des mémoires) 100
MODE CALCULATEUR (évalue une expression) 102
GO (lance l'exécution d'un programme) 103
BREAK (positionne un point d'arrêt) 104
CONTINUE (continue l'exécution d'un programme) 105
BREAK POINT (affiche les points d'arrêt) 106
REGISTER (affiche le contenu des registres) 107
YANK (suppression des points d'arrêt) 108
DUMP (visualise un bloc de mémoire) 109
MOVE (déplacement d'un bloc d'octets) 111
TRACE (visualise l'état des registres et l'instruction en cours) 113
WRITE (visualise l'état des registres et l'instruction en cours du programme principal) 113
PRINT (commute l'affichage sur l'imprimante) 114
SAVE (sauvegarde du programme objet) 115
LOAD (chargement d'un programme objet) 116

VERIFY (compare le contenu d'un fichier avec la mémoire) 117
QUIT (retour sous contrôle de menu) 118
EXIT (passage sous contrôle de l'éditeur) 119

5 - Gestion de fichiers 120
DIRECTORY (catalogue d'une disquette) 120
COPY (copie d'un fichier) 122
RENAME (change le nom d'un fichier) 123
KILL (suppression d'un fichier) 124
FORMAT (initialisation d'une disquette) 125
PRINTER COLUMNS (déclare le nombre de colonnes de l'imprimante) 126
Format d'un fichier objet 127

Annexe A : Points d'entrée du moniteur MO5 128

Annexe B : Microprocesseur 68Ø9 (instructions et adressage) . 158

Annexe C : Table ASCII 181

Annexe D : Messages d'erreur 183

Annexe E : Programmes types 186

Conseils bibliographiques 191

AVANT-PROPOS

Vous venez d'acquérir ou de vous faire offrir une cartouche ASSEMBLEUR MO5.
Ce module est un nouvel atout pour votre ordinateur.
Avec cette cartouche "Module Langage 68Ø9" vous allez pouvoir écrire des programmes dans un langage aussi près que possible de la machine.
Avant d'aller plus avant, il faut que vous compreniez que le « cœur » ou le « cerveau » de votre MO5 est un microprocesseur 68Ø9.
Le 68Ø9 ne comprend qu'un seul langage, le sien, c'est le langage machine.
Le langage machine est constitué d'une succession de codes binaires, exprimés en hexadécimal, dont le regroupement, par paquets de un, deux, trois, quatre ou cinq valeurs, est appelé instruction. La succession d'instructions constitue le programme machine ou programme objet.
Les programmes écrits en langages évolués (BASIC, FORTH, LOGO...) pour être exécutables, c'est-à-dire compréhensibles par le 68Ø9, doivent être traduits en programme machine.
Pour cette tâche on peut retenir deux solutions :
1 - Traduire le programme évolué, en totalité, avant de l'exécuter en machine. Ce travail est effectué par un autre programme que l'on appelle un compilateur. On dit alors que le programme est compilé puis exécuté.
2 - Traduire le programme ligne par ligne ou instruction par instruction avant l'exécution en machine des codes binaires correspondants.
On dit alors que le langage évolué est interprété.
Le BASIC, le FORTH, le LOGO du MO5 sont des langages interprétés.
La première solution implique un traitement de traduction avant l'exécution du programme. Mais une fois ce travail effectué, une nouvelle exécution ne nécessite plus de compilation.

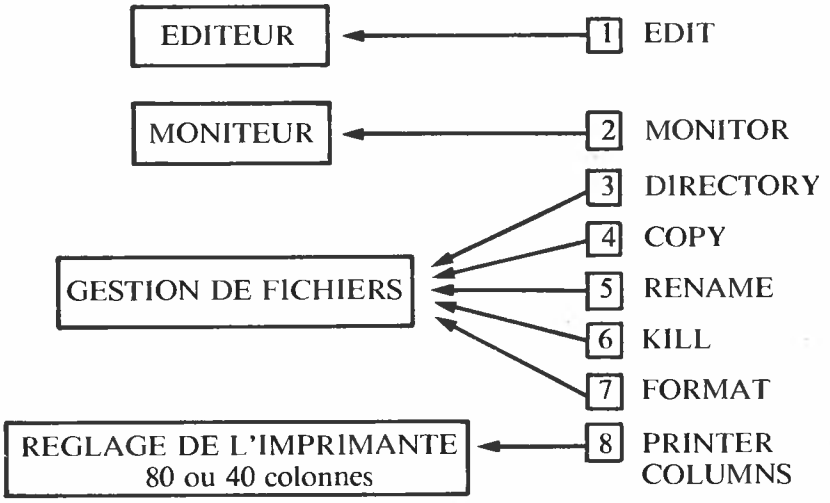
La seconde solution nécessite une traduction de chaque instruction avant son exécution, d'où une vitesse d'exécution réduite.
C'est principalement l'amélioration de ce paramètre que vous propose votre ASSEMBLEUR.
La vitesse d'exécution d'une instruction est un paramètre important. L'augmentation de la vitesse d'exécution va vous permettre d'écrire des programmes (d'animation par exemple) qui n'auraient aucun sens dans un langage interprété car leur exécution serait trop lente.
Le rôle de l'ASSEMBLEUR est de traduire la représentation mnémonique des instructions en leur équivalent binaire qui peut comporter entre un et cinq octets. Un octet est un mot de huit bits. Ce code binaire résultant, appelé **programme objet**, est directement exécutable par le microprocesseur. Le programme écrit en mnémonique est appelé **programme source**.
Cette tâche, que l'on appelle « assemblage », est réalisée par l'ASSEMBLEUR.
La cartouche « MEMO5 » contient également un ÉDITEUR destiné à écrire les programmes source, ainsi qu'un MONITEUR qui vous permettra de vérifier le fonctionnement du programme objet avant de l'exécuter ou de l'appeler, éventuellement sous contrôle d'un autre langage.
Le langage assembleur reste très près de la machine. Il dépend des limites du microprocesseur employé. C'est pourquoi nous pensons qu'il est nécessaire de bien comprendre la structure interne ainsi que le mode d'adressage du microprocesseur utilisé pour programmer de façon efficace en assembleur.

Cet ouvrage, qui est un manuel de base, et non pas un cours sur l'assembleur, décrit d'une façon détaillée les diverses commandes à votre disposition.
Chaque commande est appliquée, à titre d'exemple, sur un programme type utilisé dans l'ensemble du livre.
En annexe vous trouvez les points d'entrée du moniteur système, les instructions et le mode d'adressage du 6809.
Enfin un conseil : soyez patient ! L'écriture d'un programme en assembleur est un peu plus compliquée qu'en BASIC, mais va vous ouvrir des possibilités que vous ne soupçonnez pas.

BONNE PROGRAMMATION (en assembleur !!).

CHAPITRE 1
ORGANISATION DE LA CARTOUCHE ASSEMBLEUR

La cartouche Assembleur traite les programmes source écrits en langage d'assemblage 68Ø9. Elle traduit les instructions source en un programme objet compatible avec le microprocesseur 68Ø9.
Le MENU donne accès à quatre blocs de programmes qui sont : ÉDITEUR, MONITEUR, GESTION DE FICHIERS, RÉGLAGE IMPRIMANTE.
Leurs points d'entrée par le clavier sont les suivants :

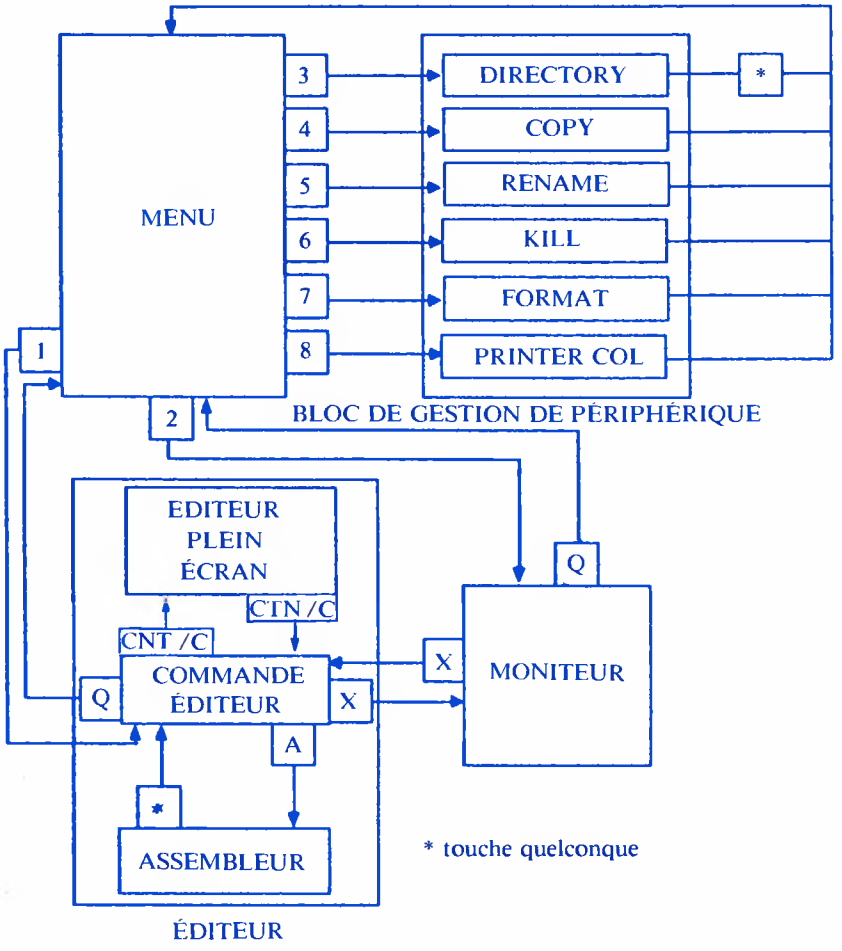


La ligne de commande **EDIT** permet l'accès à l'éditeur et à la fonction **ASSEMBLEUR** (résidant dans l'éditeur).
La ligne de commandes **MONITOR** permet l'accès au moniteur.
Le bloc GESTION DE FICHIERS est constitué de cinq routines accessibles par le MENU.

— **DIRECTORY** (catalogue du disque)
— **COPY** (copie d'un fichier)
— **RENAME** (renomme un fichier)
— **KILL** (efface un fichier)
— **FORMAT** (initialise un disque)

La ligne de commande **PRINTER COLUMNS** permet de déclarer la largeur du listing sur l'imprimante.

1.1 Architecture



1.2 Menu

Le choix dans MENU est le suivant :

1 EDIT	5 RENAME
5 MONITOR	5 KILL
5 DIRECTORY	5 FORMAT
4 COPY	5 PRINTER COLUMNS

L'appel d'une commande de MENU se fait en tapant le numéro correspondant.
En bas de l'écran, la ligne rouge est la zone commentaires de MENU, destinée à recevoir les messages d'erreur ou les questions à destination de l'utilisateur.
Exemple : "Are you sure Y/N?"
(Êtes-vous sûr O/N ?)
(Pour valider l'effacement d'un fichier)

ENTREE valide les commandes de **MENU**.
CNT C permet l'annulation d'une commande.

COPY, RENAME, KILL nécessitent, au moins, un descripteur de fichier avant la validation de la commande. La présence d'un descripteur n'est pas obligatoire pour entrer dans **MONITOR** et **EDIT**.

Le chiffre correspondant à votre choix sur MENU fait apparaître un carré bleu (curseur), au début d'une zone jaune délimitant le format (c'est-à-dire la longueur) du descripteur de fichier.

EDIT et **MONITOR** appellent leur écran spécifique.
EDIT copie le descripteur de fichier entré (complété éventuellement du suffixe et du périphérique) sur la ligne commande éditeur.

1.3 Descripteur de fichier

La syntaxe générale des descripteurs de fichiers est la suivante :
<descripteur de fichier> = <périphérique> : <nom du fichier>
Comme <périphérique> on peut avoir :

C : pour le lecteur-enregistreur de programmes (LEP)
0 : pour le premier lecteur de disquette
1 : pour le second lecteur de disquettes
2 : pour le troisième lecteur de disquettes
3 : pour le quatrième lecteur de disquettes

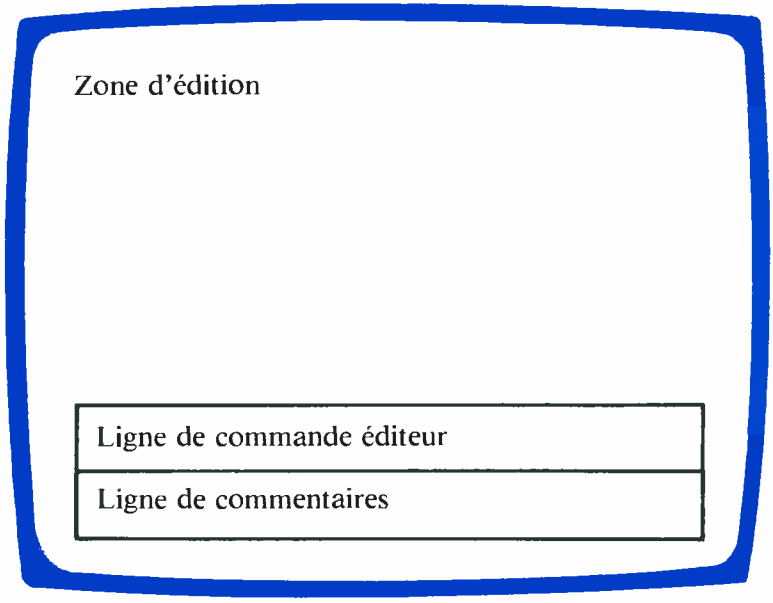
Par défaut, (si <périphérique> est absent dans le <descripteur de fichier>) le système prend le LEP comme périphérique, si aucun lecteur de disquettes n'est présent.
Si un lecteur de disquettes est présent, le lecteur Ø est pris par défaut.

Nom de fichier : le <nom de fichier> est constitué du nom lui-même et d'un suffixe optionnel.
<nom de fichier> = <nom>. <suffixe>
<nom> est composé de un à huit caractères au maximum.
<suffixe> est optionnel et comporte un à trois caractères.
<nom> et <suffixe> doivent être séparés par un point.
Par défaut (en l'absence d'un autre suffixe), le système ajoute
.ASM pour les fichiers source
.BIN pour les fichiers objet
Exemples de descripteurs de fichier :

2 : ASSEMBLE.BIN 1 : ESSAI. MO5
0 : TRAIN.BUS C : PARIS.V

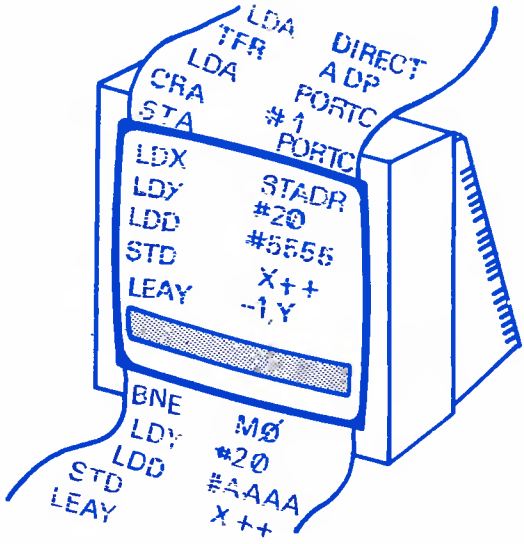
CHAPITRE 2
L'ÉDITEUR

L'éditeur permet l'écriture et l'assemblage d'un programme source. Il permet, en phase d'édition, la correction des erreurs, l'insertion de nouvelles lignes, la recherche d'étiquettes, le transfert de blocs de programme d'un/vers un périphérique etc.
L'éditeur utilise l'écran dans sa totalité et accepte, en entrée, des lignes de programmes source ou des commandes. Il génère en sortie, après assemblage, le programme objet. Les lignes de programmes sont entrées en mode ÉDITION. Les commandes sont entrées en mode COMMANDES ÉDITEUR.
L'écran sous contrôle de l'éditeur est divisé en trois zones :



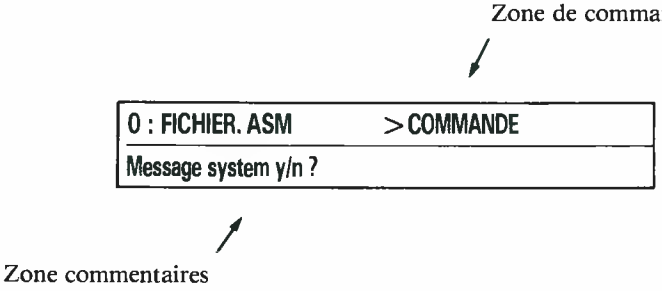
A - Zone d'ÉDITION : où l'éditeur plein écran est disponible (de la ligne Ø à la ligne 23, caractères bleus sur fond noir). La partie de programme affichée à l'écran est une fenêtre sur le programme source résidant dans la mémoire.

écran



B - Zone de COMMANDE ÉDITEUR : (ligne 24 - Fond jaune, texte bleu). La zone de commande éditeur est divisée en deux parties par « > ».
— La partie gauche affiche le descripteur de fichier en cours de traitement.
— La partie droite reçoit les commandes éditeur

C - Zone COMMENTAIRES : (ligne 25 - Fond rouge, écriture noire). La zone commentaires est destinée à recevoir les questions à destination de l'utilisateur ou les messages d'erreur.
Exemple : "Printer ready Y/N" ou "Bad command".



Accès à l'éditeur : la commande de MENU 1 active le bloc ÉDITEUR.

A - Sans descripteur de fichier : édition d'un nouveau programme ou d'un programme déjà résidant en mémoire.

1	EDIT		ENTREE

L'écran de l'ÉDITEUR est généré et les vingt-trois premières lignes du programme source, éventuellement résidant en mémoire, sont affichées. Le curseur se place en Ø,Ø (en haut et à gauche). Le système attend une entrée sur la ligne de commande éditeur.

B - Avec descripteur de fichier : chargement et édition d'un programme.

	4	EDIT	C: ASSEMBLE.ASM	ENTREE
ou	1	EDIT	ASSEMBLE.ASM	ENTREE
ou	1	EDIT	ASSEMBLE	ENTREE

Note : C et . ASM sont facultatifs et sont pris par défaut (si on travaille avec un LEP).

L'écran de l'éditeur est généré comme ci-dessus.
Le descripteur de fichier, complété éventuellement du périphérique et du suffixe, est affiché dans la partie gauche de la ligne de commande éditeur.
Exemple :

```
C: ASSEMBLE.ASM      >
```

Le programme se charge, le curseur se place en Ø,Ø et les vingt-trois premières lignes du programme source s'affichent sur l'écran. Une entrée est attendue sur la ligne de commande éditeur.

Q ENTREE Redonne la main à MENU

x ENTREE Donne accès au moniteur

CNT C Permet le passage du mode ÉDITION en mode COMMANDE ÉDITEUR et réciproquement.

Note : seuls les fichiers source peuvent être chargés par l'éditeur. Un appel de fichier binaire sous éditeur provoque l'erreur "File format error".

Ecriture d'un programme source : l'écriture d'un programme source est réalisée sous contrôle de l'éditeur, en mode ÉDITION.
Pour placer le système sous contrôle de l'éditeur, vous disposez de deux commandes.
— A partir de MENU : Tapez 1 puis ENTREE
— A partir du MONITEUR : Tapez x puis ENTREE
Vous êtes maintenant sous contrôle de l'éditeur.
Un carré bleu (curseur) dans la zone de commande éditeur indique que vous êtes en mode COMMANDE ÉDITEUR.
La commande CNT C permet le passage du mode COMMANDE ÉDITEUR au mode ÉDITION
Une nouvelle commande CNT C redonne la main au mode COMMANDE ÉDITEUR.
Nous vous invitons à essayer cette commande.

Remarquez que le carré bleu (curseur) disparaît de la ligne de commande en mode ÉDITION. Cela vous permettra de repérer « d'un coup d'œil » le mode en cours dans l'éditeur.
Placez-vous en mode ÉDITION (carré bleu absent) : le curseur est actif dans la zone d'édition.
Chaque ligne de programme source est formée de quatre champs :
Une étiquette (ou le caractère (*) pour les lignes de commentaires), un code opération, un opérande et un commentaire.

Étiquette	Code opération	Opérande	Commentaire
Ø 6	7 13	14 2Ø	22 39

Le champ étiquette

Le champ étiquette est le premier champ d'une ligne de programme source. Il est défini entre les colonnes Ø et 6 et il peut prendre l'une des formes suivantes :

1. Un astérisque (*) placé en tête du champ étiquette signifie que le reste de la ligne du programme source est un commentaire. Les commentaires sont ignorés par l'assembleur, et ne sont présents dans le programme source que pour informer l'utilisateur.

2. Un caractère en tête de la ligne signifie que celle-ci possède une étiquette. Les caractères sont les lettres majuscules de A à Z et les chiffres de Ø à 9. Les caractères spéciaux : point (.), dollar ($) etc. ne sont pas admis. Les étiquettes se composent de un à six caractères. Le premier doit être obligatoirement un caractère alphabétique. Certaines étiquettes sont réservées à l'assembleur et ne doivent pas être utilisées dans le champ étiquette. Dans le cas où on ne respecterait pas cette règle, une erreur serait générée. Les étiquettes réservées sont A, B, D, X, Y, CC, DP, PC, S, U qui représentent les registres du 68Ø9.

3. Un espace en tête de la ligne source indique que le champ étiquette est vide. La ligne n'a alors pas d'étiquette et ne peut être un commentaire.

Une étiquette ne peut être présente qu'une seule fois dans le champ étiquette, sauf lorsqu'elle est utilisée avec la directive d'assemblage **SET**. Si une étiquette est utilisée plusieurs fois, chaque passage sur cette étiquette sera signalé par l'erreur "Multiply defined symbol".

Le champ code opération

Le champ code opération se trouve après le champ étiquette entre les colonnes 7 et 13. Il doit être précédé d'au moins un espace et contenir le mnémonique d'une instruction 68Ø9 ou une directive d'assemblage. Il est composé uniquement des lettres majuscules de A à Z. Le contenu du champ code opération peut donc être de deux types :
— Mnémoniques : correspondant aux instructions machine du microprocesseur 68Ø9.
— Directives d'assemblage : des codes opération reconnus par l'assembleur, agissant plutôt sur le processus d'assemblage et qui ne sont pas traduites en programme objet.

Le champ opérande

Le champ opérande suit le champ code opération. Il se situe entre les colonnes 14 et 2Ø mais peut être étendu jusqu'à la colonne 39. Dans tous les cas, un espace doit être utilisé comme séparateur entre le champ opérande et le champ commentaire. L'interprétation du champ opérande dépend du contenu du champ code opération. Il peut contenir une étiquette, un symbole, une expression, un nombre ou une combinaison des quatre. Il sert également à spécifier le mode d'adressage quand le champ code opération contient un mnémonique.

Les étiquettes et les symboles

Les étiquettes et les symboles sont constitués de six caractères au plus et commencent par une lettre. Une valeur entière comprise entre Ø et 65535 (seize bits) leur est associée et vient les remplacer dans l'évaluation d'une expression qui contient les étiquettes ou les symboles spécifiés.

Certains symboles sont spécifiques et ne peuvent être utilisés que dans les expressions. Ce sont :
(*) (astérisque) ou (.) (point), qui représente la valeur courante du compteur de programme (PC).
ENDMEM, qui définit la fin de la mémoire utilisateur.

Les expressions

Les expressions sont composées d'une suite de symboles, de nombres, d'opérateurs logiques ou arithmétiques et de parenthèses. Les expressions dans le champ opérande servent à spécifier une valeur. Les règles de la logique et de l'arithmétique s'appliquent aux expressions.

Exemple :

ENDMEM-$4000 **DEPART+$30** ***<-8**

... sont des expressions

Les opérateurs

Les opérateurs dans une expression peuvent être de deux types : logiques ou arithmétiques, et ils sont hiérarchisés de un à cinq. Les opérations d'un niveau supérieur seront effectuées avant les opérations de niveau inférieur. Les opérations de même niveau hiérarchique seront évaluées de la gauche vers la droite. Les résultats intermédiaires et de l'expression sont arrondis à une valeur entière. Les opérateurs peuvent agir sur les nombres et les symboles.

OPÉRATEUR	SYMBOLES	NIVEAU HIÉRARCHIQUE
Les opérateurs arithmétiques		
DIVISION	.DIV. ou /	5
MULTIPLICATION	*	5
ADDITION	+	2
SOUSTRACTION	–	2
MODULO	.MOD.	5
Les opérateurs logiques		
ET logique	.AND. ou &	4
OU inclusif	.OR. ou !	3
Ou exclusif	.XOR.	3
Décalage à droite de n	<-n	5
Décalage à gauche de n	< [+] n	5
EGALITE*	.EQU. ou =	1
INEGALITE**	.NEQ.	1

* La condition vraie rend – 1 ($FFFF ou $FF).
** La condition fausse rend Ø.

Les nombres

Les nombres représentent des données qui ne varient pas avec l'exécution du programme. Les nombres peuvent être représentés dans trois bases : décimale (base 1Ø), octale (base 8), hexadécimale (base 16). La représentation binaire n'est pas admise. Un nombre sera reconnu comme étant écrit dans une base donnée par la présence d'un préfixe ou d'un suffixe (voir la table ci-dessous). **Sans préfixe ni suffixe, un nombre sera interprété comme étant un nombre décimal par l'assembleur.**

BASE	PRÉFIXE	SUFFIXE	PLAGE
OCTALE	@	Q ou O	Ø à 177777
DÉCIMALE	&	T	Ø à 65535
HEXADÉCIMALE	$	H	Ø à FFFF

Exemples : 2ØØØ H = 8192T &5ØØØ = $1388

Le mode d'adressage

ADRESSAGE	SYNTAXE
Implicite ou inhérent	Pas d'opérande
Direct	< <expression>
Etendu	> <expression>
Immédiat	# <expression>
Etendu indirect	[<expression>]
Indexé	(expression),R
Indexé (8 bits)	< <expression>,R
Indexé (16 bits)	> <expression>,R
Indexé indirect	[<expression>]
Indexé indirect (8 bits)	<[<expression>,R]
Indexé indirect(16 bits)	>[<expression>,R]
Post-incrémenté de 1	,R+
Post-incrémenté de 2	,R++
Post-incrémenté indirect	[,R++]
Pré-décrémenté de 1	,−R
Pré-décrémenté de 2	,−−R
Pré-décrémenté indirect	[,−−R]

Le champ commentaire

Le champ commentaire est le dernier d'une ligne de programme source. Il peut commencer à partir du premier espace après le champ opérande. C'est un champ optionnel : il sert à la documentation du programme et est seulement imprimé sur le listing mais n'est pas traduit en langage machine.
Le passage d'un champ à l'autre se fait automatiquement par [ ] (barre d'espacement), si vous n'êtes pas dans le champ commentaire.
Dans le champ commentaire, l'espace reprend sa fonction normale.
Sur les champs étiquette, code opération, opérande, les caractères minuscules sont automatiquement remplacés par les majuscules équivalentes. Ainsi vous pouvez taper indifféremment en majuscules ou en minuscules.
Le passage en colonne 37 active la note « DO » pour signaler à l'utilisateur que la fin de la ligne approche (col. 39).

Les programmes d'essai

Nous vous proposons d'écrire deux programmes d'essai :
« MOIRAGE » et « EQUATE »
qui nous serviront de références dans la suite de cet ouvrage. Pour écrire ces programmes vous devrez utiliser les commandes d'édition qui sont des « outils » destinés à écrire et corriger les programmes source. Nous vous invitons à essayer ces « outils » en écrivant les programmes d'essai ci-dessous.
En mode ÉDITION, vous disposez d'un certain nombre de commandes qui sont accessibles par les touches de contrôle du clavier du MO5.

Les touches suivantes sont utilisables :

[↑] [↓]	Décalage en haut ou en bas du curseur dans le texte édité.
[→] [←]	Décalage à droite ou à gauche du curseur sur une ligne de texte édité.
[EFF] [INS]	Insertion ou effacement d'un caractère dans une ligne de texte.
[RAZ]	Décalage d'une page vers le bas ou vers le haut.
[ENTREE]	Insertion d'une nouvelle ligne dans le texte.
[ ]	Tabulation automatique ou espace.

Programme « EQUATE »

« EQUATE » est un programmes de définition des codes d'entrée du moniteur système (MO5) des adresses et des codes ASCII nécessaires au fonctionnement du programme « MOIRAGE ».

```
*
* POINTS D'ENTREE DU MONITEUR MO5
*
RETCAR EQU     $00      Main cartouche ASM
PUTCH  EQU     $02      Affichage
GETCH  EQU     $0A      clavier
KTST   EQU     $0C      Test clavier
DRAW   EQU     $0E      Ligne
PLOT   EQU     $10      Point
CARAC  EQU     $12      Caractere
RSCONT EQU     $24      RS-232
K7CONT EQU     $20      Lecteur de k7
GETLP  EQU     $18      Crayon optique
LPINT  EQU     $16      Interrupteur.
NOTE   EQU     $1E      Musique
GETPT  EQU     $14      Lecture Point
GETSC  EQU     $1A      Lecture caractere
JOYSTK EQU     $1C      Manche a balais
DKCONT EQU     $26      Controleur disque
MIMECO EQU     $04      Memoire couleur
MIMEFO EQU     $06      Memoire forme
BIP    EQU     $08      Emission d'un BIP
MOTLEP EQU     $22      M/A moteur LEP
*
* Adresses Physiques
*
PORTA  EQU     $A7C0    Le bit 0 du Port A
* controle l'acces a la memoire ecran
* 1=Points ; 0=couleur
STADR  EQU     $0000    Debut de l'ecran
ENDADR EQU     $1F40    Fin de l'ecran +1
*
* CODES ASCII
*
US     EQU     $1F      UNIT SEPARATOR
FF     EQU     $C       FORM FEED
EOT    EQU     4        END OF TRANSM
ESC    EQU     $1B      ESCAPE
CR     EQU     $D       CARRIAGE RETURN
```

Sauvegarde d'« EQUATE »

La sauvegarde d'« EQUATE » est obtenue en mode COMMANDE ÉDITEUR par l'utilisation de la commande **S (SAVE)** :

```
>SØ : EQUATE.ASM
```

pour la sauvegarde sur la disquette qui se trouve dans l'unité Ø.

```
>SC : EQUATE.ASM
```

pour la sauvegarde sur le lecteur de cassettes.
Le système vous pose la question :

```
Cassette Ready Y/N?
```

Tapez **Y**
EQUATE est sauvegardé et le nom du fichier créé est affiché sur la ligne de commande.

```
C : EQUATE.ASM          >
```

La vérification du fichier sauvegardé sur la cassette peut être obtenue avec la commande **V (VERIFY)**

Pour cela procédez comme suit :

— Entrez la commande **V**

```
C : EQUATE.ASM          >V
```

Le système vous répond en vous posant la question :

```
Cassette Ready Y/N?
```

— Rembobinez la cassette
— Répondez **Y**
Si « EQUATE » est sauvegardé avec des erreurs le système affichera :

```
Verification Error
```

Dans ce cas il faut refaire la sauvegarde d'« EQUATE ».
La commande **N (NEW)**

<descripteur>	>N
Are you sure Y/N ?	

efface le programme édité (« EQUATE »).

Programme « MOIRAGE »

« MOIRAGE » est un programme d'écriture, en damier, des groupes de points ligne (GPL) en mode forme. Notez que « MOIRAGE » n'intervient pas sur la mémoire couleur qui reste ce qu'elle était.

```
**********MOIRAGE**********
* Programme de moirage de la memoire
* Point
        ORG     ENDMEM-$400 1K Reserve
DIRECT  EQU     *<-8    Page 0
        SETDP   DIRECT
        TITLE   Balayage Ecran
        INCLUD EQUATE       fichier con-
* tenant les Principaux codes d'entree
* du Moniteur MO5

        PAGE
START   LDA     PORTA   Mise en memoire
* Points:Par mise a 1 du bit 0 du Port A

        ORA     #$1
        STA     PORTA
        LDX     #STADR  Adresse de debut
* de l'ecran
M2      LDY     #20     Compteur colonne:
* on affiche 20 fois 2 octets,soit
* 40 octets par ligne

        LDD     #$5555  Moirage
* alternance de 1 et 0 sur la ligne
M0      STD     ,X++    Charger l'ecran
        LEAY    -1,Y
        BNE     M0      Repete 20 fois
        LDY     #20     Ligne suivante
        LDD     #$AAAA  Le motif est inver
* se pour decaler les 1 et les 0 d'une
* ligne a l'autre et obtenir un moirage
```

```
M1      STD     ,X++
        LEAY    -1,Y
        BNE     M1          Toujours 20 fois
        CMPX    #ENDADR Fin d'ecran
        BLS     M2          Sinon on recom
* mence 2 lignes a motifs alternes
        STOP                Retour au moniteur
        END     START
```

---

* La commande INCLUD ne fonctionne qu'en version disquette. Avec un lecteur de cassette, il faut retaper à la place le contenu du fichier "EQUATE".

Sauvegarde de « MOIRAGE »

La sauvegarde de « MOIRAGE » est obtenue par :

```
>SØ : MOIRAGE.ASM
```

ou

```
SC : MOIRAGE.ASM
```

puis (facultatif)

```
Ø : MOIRAGE.ASM      > V
```

ou

```
C : MOIRAGE. ASM     > V
```

2.1 Commandes d'édition

Les commandes d'édition sont des commandes actives en mode ÉDITION dans le champ du programme.

Déplacement du curseur d'une position vers le haut

Commande : La touche ↑

Effet : La commande ↑ déplace le curseur d'une ligne vers le haut de l'écran. Si le curseur est positionné en ligne Ø, le reste du texte visible à l'écran est repoussé d'une position vers le bas, et la ligne qui précédait est affichée en haut de l'écran. Si la ligne Ø est la première du programme source résidant dans l'éditeur, une ligne blanche est ajoutée au début du programme.

Exemple :

```
**********MOIRAGE**********
* Programme de moirage de la memoire
* point


        LDD     #$5555  Moirage
* alternance de 1 et Ø sur la ligne
MØ      STD     ,X++    Charger l'ecran
```

Tapez ↑

```
**********MOIRAGE**********
* Programme de moirage de la memoire
* point


        LDD     #$5555  Moirage
* alternance de 1 et Ø sur la ligne
```

Déplacement du curseur d'une position vers le bas

Commande : La touche ↓

Effet : La commande ↓ déplace la curseur d'une ligne vers le bas de l'écran. Le curseur positionné en ligne 23 provoque un défilement de la zone d'édition vers le haut de l'écran et affiche la ligne suivante en ligne 23. Si le curseur est positionné sur la dernière ligne du programme source résidant, après le déplacement le curseur pointera sur une ligne blanche qui ne sera pas ajoutée au texte du programme.

Exemple :

```
        STA     PORTA
        LDX     #STADR  Adresse de debut
* de l'ecran
M2      LDY     #20     Compteur colonne
* On affiche 20 fois 2 octets,soit

        CMPX    #ENDADR Fin d'ecran
        BLS     M2      Sinon on recom
* mence 2 lignes a motifs alternes
        STOP            Retour au moniteur
        END     START
```

Tapez : ↓

```
        LDX     #STADR  Adresse de debut
* de l'ecran
M2      LDY     #20     Compteur colonne
* On affiche 20 fois 2 octets,soit
* 40 octets par ligne

        BLS     M2      Sinon on recom
* mence 2 lignes a motifs alternes
        STOP            Retour au moniteur
        END     START
```

Déplacement du curseur d'une position vers la gauche

Commande : La touche ←

Effet : La commande ← déplace le curseur d'un caractère vers la gauche de l'écran. Si le curseur est positionné sur la colonne Ø, une commande ← le localisera à la fin de la ligne précédente. Si le curseur est situé en ligne Ø, un déplacement sur la colonne Ø visualisera une nouvelle ligne en haut de l'écran et positionnera le curseur en ligne Ø et colonne 39.

Exemple :

```
* de l'ecran
M2      LOOP    #20      Compteur colonne:
```

Tapez ←

```
* de l'ecran
M2      LOOP    #20      Compteur colonne:
```

Déplacement du curseur d'une position vers la droite

Commande : La touche →

Effet : La commande → déplace le curseur d'un caractère vers la droite de l'écran. Si le curseur est positionné sur la colonne 39, une commande → le localisera au début de la ligne suivante. Si le curseur est situé en ligne 23, un déplacement sur la colonne 39 visualisera une nouvelle ligne en bas de l'écran et positionnera le curseur en ligne 23 et colonne Ø.

Exemple :

```
        INCLUD EQUATE        fichier con-
* tenant les PrinciPaux codes d'entree
```

Tapez

```
        INCLUD EQUATE        fichier con-
* tenant les PrinciPaux codes d'entree
```

Espace

Commande : La touche [ ] (barre d'espacement)

Effet : La commande [ ] (espace) a plusieurs fonctions :
— Tabulation automatique dans l'écriture d'une ligne de programme source, sauf sur le champ commentaire.
— Un espace dans les zones suivantes :
* ligne de commande éditeur
* lignes de commentaires
* champ commentaire d'une ligne de programme source.

Exemple :

Tapez [ ]
```
DIRECT EQU     *<-8     Page 0
```
Tapez [ ]
```
DIRECT EQU     *<-8     Page 0
```
Tapez [ ]
```
DIRECT EQU     *<-8     Page 0
```
Tapez [ ]
```
DIRECT EQU     *<-8     Page 0
```
Tapez [ ]
```
DIRECT EQU     *<-8     Page 0
```
Tapez [ ]
```
DIRECT EQU     *<-8     Page 0
```

Entrée

Commande : A partir du clavier, ENTREE

Effet : La commande ENTREE , activée dans la zone d'ÉDITION, provoque un saut au début de la ligne suivante en insérant une nouvelle ligne et décale le reste du texte vers le bas. En ligne 23 le texte est poussé vers le haut et le curseur reste positionné au début de la ligne 23. Sur la ligne de commande éditeur la commande : ENTREE valide la/les commandes précédentes.

Exemple :

```
M2      LDY     #20      Compteur colonne:
* on affiche 20 fois 2 octets,soit
```

Tapez ENTREE

```
M2      LDY     #20      Compteur colonne:

* on affiche 20 fois 2 octets,soit
```

Insertion d'un caractère

Commande : La touche INS

Effet : La commande INS permet l'insertion d'un nouveau caractère à partir de la position du curseur, en décalant le reste du texte vers la droite à l'intérieur du champ défini. L'insertion dans un champ d'une ligne de programme source est effectuée jusqu'à ce que le caractère qui a été initialement pointé par le curseur, atteigne le champ suivant.
Quand le caractère pointé a atteint le champ suivant une nouvelle commande d'insertion INS est sans effet.
Tout caractère décalé au-delà de la colonne 39 est définitivement perdu.

Exemple :

```
        START   LDA     PORTA   Mise en memoire
```

Tapez INS

```
        S TART  LDA     PORTA   Mise en memoire
```

Tapez INS

```
        S  TARTLDA      PORTA   Mise en memoire
```

Tapez INS

```
        S   TARTLDA     PORTA   Mise en memoire
```

Tapez INS

```
        S    TARTLDA    PORTA   Mise en memoire
```

Tapez INS

```
        S     TARTLDA PORTA     Mise en memoire
```

Tapez INS

```
        S      TARTLDAPORTA     Mise en memoire
```

Tapez INS

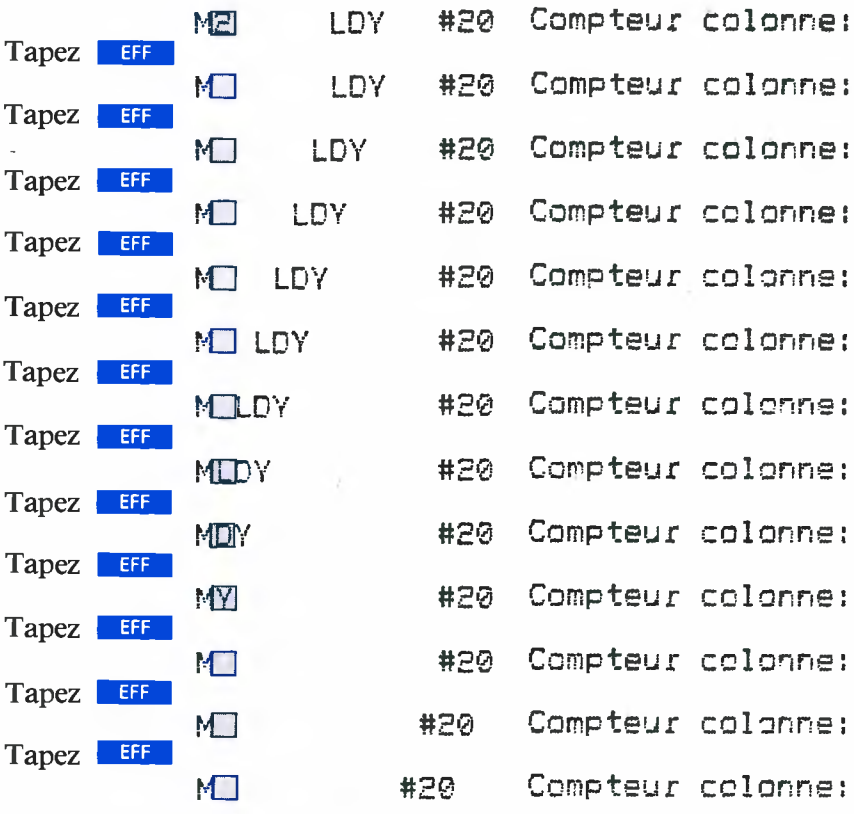
```
        S      TARTLDAPORTA     Mise en memoire
```

Effacement d'un caractère

Commande : La touche EFF

Effet : La commande EFF efface le caractère pointé par le curseur et décale vers la gauche les caractères à droite de la position pointée.
Sur une ligne de programme source, si le champ pointé est vide à droite du curseur, le contenu du champ suivant, ayant au moins un caractère, est décalé vers la gauche jusqu'à la position du curseur.
Si le champ suivant le champ pointé est vide à son tour, c'est le champ +2 qui est attiré vers le champ pointé, puis le champ +3, etc.
Sur une ligne vide, la commande EFF est sans effet.

Exemple :



Descente d'une page

Commande : La touche RAZ

Effet : La commande de descente de page RAZ dans l'éditeur permet d'afficher à l'écran en zone d'édition les vingt-trois lignes qui suivent la page courante.
Si la dernière ligne du programme source résidant est affichée sur l'écran, la commande RAZ est sans effet. La position du curseur sur l'écran n'est pas modifiée par la commande.

Exemple :

```
**********MOIRAGE**********
* Programme de moirage de la memoire
* Point
          ORG     ENDMEM-$400 1K Reserve

* 40 octets par ligne
          LDD     #$5555  Moirage
* alternance de 1 et 0 sur la ligne
M0        STD     ,X++    Charger l'ecran
```

Tapez RAZ

```
          STA     PORTA
          LDX     #STADR  Adresse de debut
* de l'ecran
M2        LDY     #20     Compteur colonne:

          BLS     M2      Sinon on recom
* mence 2 lignes a motifs alternes
          STOP            Retour au moniteur
          END     START
```

Remontée d'une page

Commande : La touche

Effet : La commande de remontée d'une page dans l'éditeur permet d'afficher à l'écran, en zone d'édition, les vingt-trois lignes qui précédent la page courante.
Si la première ligne du programme source résidant est affichée sur l'écran, la commande est sans effet. La position du curseur sur l'écran n'est pas modifiée par la commande

Exemple :

```
        STA     PORTA
        LDX     #STADR  Adresse de debut
* de l'écran
M2      LDY     #20     Compteur colonne:
```

```
        BLS     M2      Sinon on recom
* mence 2 lignes a motifs alternes
        STOP            Retour au moniteur
        END     START
```

Tapez

```
**********MOIRAGE**********
* Programme de moirage de la memoire
* Point
        ORG     ENDMEM-$400 1K Reserve
```

```
* 40 octets par ligne
        LDD     #$5555  Moirage
* alternance de 1 et 0 sur la ligne
M0      STD     ,X++    Charger l'ecran
```

Note : les autres commandes disponibles en édition exigent des frappes doubles du type CNT et une autre touche appuyée en même temps.
Ces commandes seront représentées dans la suite par CNT ?

Exemple : CNT X ou CNT D etc.

Effacement d'une ligne dans la zone d'ÉDITION

Commande : Les touches CNT X

Effet : La commande CNT X efface la ligne courante à partir de la position du curseur. Si le curseur est positionné en colonne Ø, la commande CNT X efface la totalité de la ligne, remonte le reste du texte vers le haut de l'écran et repositionne le curseur au début de la ligne suivante.

Exemple 1

```
        ORG     ENDMEN-$400 1K Reserve
DIRECT  EQU     *<-8        Page 0
        SETDP   DIRECT
```

Tapez CNT X

```
        ORG
DIRECT  EQU     *<-8        Page 0
        SETDP   DIRECT
```

Exemple 2

```
        DRG     ENDMEN-$400 1K Reserve
DIRECT  EQU     *<-8        Page 0
        SETDP   DIRECT
```

Tapez CNT X

```
        ORG     ENDMEN-$400 1K Reserve
        SETDP   DIRECT

```

Positionnement du curseur en fin de la ligne courante

Commande : Les touches CNT E

Effet : La commande CNT E positionne le curseur à la fin de la ligne courante.

Exemple :

```
START   LDA     PORTA    Mise en memoire
```

Tapez CNT E

```
START   LDA     PORTA    Mise en memoire
```

Positionnement du curseur au début de la ligne courante

Commande : Les touches CNT D

Effet : La commande CNT D positionne le curseur au début de la ligne courante.

Exemple :

```
M0      STD     ,X++     Charger l'ecran
```

Tapez CNT D

```
M0      STD     ,X++     Charger l'ecran
```

Caractères spéciaux « [» et «] »

CNT A → « [» CNT Z → «] »

Effet : Les touches CNT A et CNT Z génèrent les caractères « [» et «] ».
Ces caractères sont utilisés pour définir dans un programme source l'adressage indirect.

Exemple :

```
LDA   [$E856]
```

Affichage d'un espace en vidéo inversée

Commande : Les touches CNT S

Effet : Les touches CNT S génèrent un espace en vidéo inversée. Cet « espace inversé » est nécessaire pour remplacer la touche espace qui, dans la phase d'écriture ou de modification d'une ligne de programme source, provoque une tabulation automatique. Pour avoir un espace dans la directive d'assemblage **FCC**, il faut utiliser la commande CNT D.
Notez que pour mettre des caractères minuscules dans un **FCC**, il faut être en minuscules et utiliser le **SHIFT** pour ne pas être forcé en majuscules.

Exemple : Écriture d'une ligne contenant la directive d'assemblage **FCC**, en utilisant l'espace normal :

```
FCC/MO5
```

Tapez un espace avec la barre d'espacement.

```
FCC/MO5      THOMSON/
```

Écriture de la même ligne en utilisant l'espace inversé.

```
FCC/MO5
```

Tapez CNT S

```
FCC/MO5 THOMSON/
```

Commande de changement d'état

Mode ÉDITION Mode COMMANDE ÉDITEUR

Commande : Les touches CNT D

Effet : La commande CNT C permet le passage dans l'éditeur du mode ÉDITION plein écran au mode COMMANDE ÉDITEUR, ou du mode COMMANDE ÉDITEUR au mode ÉDITION plein écran.

Le passage d'un mode à l'autre ne modifie pas la position du curseur dans la zone d'édition. En mode COMMANDE ÉDITEUR, un nouveau curseur est visualisé (carré bleu) dans la zone de commande. Les ordres dans la zone de commande sont validés par ENTREE
Après l'exécution de la commande, à l'exception de Q, X et A qui quittent l'éditeur ou CNT D qui passe en mode ÉDITION, le curseur (carré bleu) est repositionné dans la zone de commande, en attente d'une nouvelle commande.

Exemple : Voir « **Écriture d'un programme source** »

2.2 Commandes éditeur

Les commandes éditeur sont des commandes actives en mode COMMANDE ÉDITEUR dans la zone de commande.
Les commandes éditeur peuvent être des caractères seuls ou suivis par des paramètres. Les espaces entre les commandes et les paramètres sont facultatifs. La partie gauche de la zone de commande peut être occupée par un descripteur de fichier, si une commande précédente en a spécifié un. La partie droite est délimitée par >. Les caractères qui dépassent la colonne 39 sont perdus.
Les commandes INS EFF → ← conservent leur fonction normale dans la partie droite de la zone de commande, mais ne sont pas autorisées dans la partie gauche. Les commandes ↓ ↑ ↵ , RAZ ne sont actives que dans la zone d'édition. Toute commande erronée peut être annulée par CNT D .

TOP : Début du programme source résidant dans l'éditeur

Commande : Tapez T puis ENTREE

Effet : La commande T (TOP) affiche en zone d'édition les vingt-trois premières lignes du programme source résidant et positionne le curseur en haut et à gauche de l'écran (Ø,Ø).
La commande T peut être utilisée pour définir la limite haute de la zone sélectionnée (voir commande **ZONE**).

Exemple :

```
>T
```

ENTREE

```
**********MOIRAGE**********
* Programme de moirage de la memoire
* Point
        ORG     ENDMEM-$400 1K Reserve


* 40 octets par ligne
        LDD     #$5555  Moirage
* alternance de 1 et 0 sur la ligne
M0      STD     ,X++    Charger l'ecran
```

BOTTOM : Fin du programme source résidant dans l'éditeur

Commande : Tapez B puis ENTREE

Effet : La commande B (BOTTOM) affiche en zone d'édition les vingt-trois dernières lignes du programme source résidant et positionne le curseur après le dernier caractère de la dernière ligne du programme. La commande B peut être utilisée pour définir la limite basse de la zone sélectionnée (voir commande ZONE).

Exemple :

```
>B
```

ENTREE

```
        STA     PORTA
        LDX     #STADR  Adresse de debut
* de l'ecran
M2      LDY     #20     Compteur colonne:
```

```
        BLS     M2      Sinon on recom
* mence 2 lignes a motifs alternes
        STOP            Retour au moniteur
        END     START
```

OFF : Suppression de la tabulation automatique

Commande : Tapez O puis ENTREE

Effet : La commande O (OFF) :
— supprime la tabulation automatique des champs des lignes de programme source, en mode ÉDITION,
— annule, pour les touches INS et EFF, la division des lignes de programme source en champs,
— arrête le forçage automatique en caractères majuscules par le module éditeur.
Le retour à la tabulation automatique ne peut être obtenu qu'en entrant à nouveau dans l'ÉDITEUR. (ex : O puis 1).

Exemple :

```
>O
```

ENTREE

ZONE : Sélectionne une zone mémoire dans l'éditeur en vue de sa copie, de sa destruction, ou pour y effectuer des modifications

Commande : Tapez Z puis ENTREE

Effet : La commande Z (ZONE) repère, en vidéo inversée, une zone dans l'éditeur.
Cette zone est définie à partir de la ligne courante (ligne du curseur) et peut être modifiée en hauteur par les mouvements relatifs du curseur. Les commandes utilisables pour cette tâche sont :
↑ ↓ B D E RAZ ↩
Une nouvelle commande Z (ZONE) annule la zone sélectionnée.
Les commandes :
A D I L CNT C M D D X
annulent la sélection.
Les commandes :
D E D P R S V
n'affectent pas la zone sélectionnée.

Exemple :

```
START   LDA     PORTA    Mise en memoire
* Points:Par mise a 1 du bit 0 du Port A
        ORA     #1
```

```
>Z
```

ENTREE

```
START   LDA     PORTA    Mise en memoire
* Points:Par mise a 1 du bit 0 du Port A
        ORA     #1
```

↑

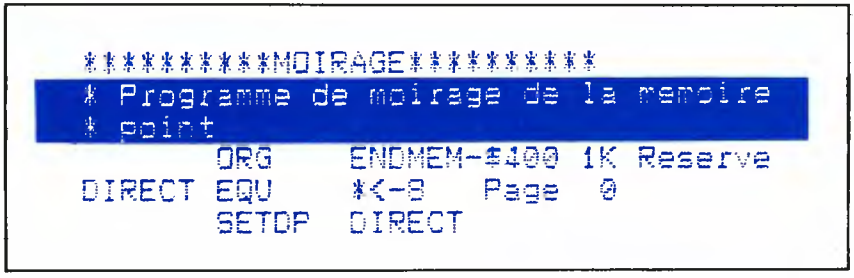
```
        PAGE
START   LDA     PORTA    Mise en memoire
* Points:Par mise a 1 du bit 0 du Port A
        ORA     #1
```

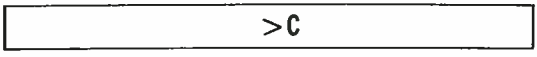
COPY : Sauvegarde dans une mémoire provisoire d'une partie du programme source

Commande : Tapez C puis ENTREE

Effet : La commande C (copy) sauvegarde la ligne courante ou la zone sélectionnée par Z dans une "mémoire de travail" en écrasant le contenu précédent. La « mémoire de travail » est effacée par les commandes X, Q et A. Les autres commandes de l'éditeur, y compris N, sont sans effet sur la « mémoire de travail ».

Exemple :





ENTREE

Les lignes...
* Programme de moirage de la mémoire
* point
... sont copiées dans la mémoire de travail
Tapez Z pour annuler la sélection de la zone.

INSERT : Insertion du contenu de la mémoire de travail dans le programme source résidant

Commande : Tapez **I** puis **ENTREE**

Effet : La commande **I** (**INSERT**) insère le contenu de la mémoire de travail au-dessus de la ligne courante.

Exemple :

Contenu de la mémoire de travail :

```
* Programme de moirage de la mémoire
* point
```

A l'écran :

```
* mence 2 lignes a motifs alternes
        STOP                Retour au moniteur
        END     START
```

```
>I
```

ENTREE

```
        BLS     M2          Sinon on recom
* mence 2 lignes a motifs alternes
* Programme de moirage de la memoire
* Point
        STOP                Retour au moniteur
        END     START
```

Remarque :

Le contenu de la mémoire de travail est inséré dans le programme source résidant, mais n'est pas écrasé pour autant. Il peut être utilisé pour une autre insertion.

DELETE : Effacement de la ligne courante ou de la zone programme sélectionnée

Commande : Tapez D puis ENTREE

Effet : La commande D (DELETE) efface la ligne courante ou la zone sélectionnée par la commande Z (ZONE). Les lignes effacées sont remplacées par les lignes qui les suivent dans le programme source. La commande D (DELETE) n'a pas d'action sur la « mémoire de travail ».

Exemple :

```
* mence 2 lignes a motifs alternes
* Programme de moirage de la memoire
* Poin
        STOP              Retour au moniteur
        END     START
```

```
>D
```

ENTREE

```
* mence 2 lignes a motifs alternes
* Programme de moirage de la memoire
        STOP              Retour au moniteur
        END     START
```

```
>D
```

ENTREE

```
* mence 2 lignes a motifs alternes
        STOP              Retour au moniteur
        END     START
```

```
>D
```

ENTREE

```
        STOP              Retour au moniteur
        END     START
```

FIND : Recherche d'une chaîne de caractères dans le programme source résidant

Commande : Tapez F/<chaîne>/ puis ENTREE
ou F puis ENTREE

Effet : La commande F/<chaîne>/ (FIND) recherche, dans le programme source résidant, la première expression /<chaîne>/ en partant de la position courante. /<chaîne>/ peut être située dans n'importe lequel des quatre champs des lignes de programme.
Dès que /<chaîne>/ est identifiée, le curseur pointe le premier caractère de la chaîne.
Si / <chaîne> / est omise dans la commande, c'est la chaîne spécifiée dans la commande F (FIND) précédente qui sera utilisée pour la recherche. La commande F (FIND) peut être utilisée avec la commande Z (ZONE) active. Dans ce cas la recherche étend la zone sélectionnée jusqu'à la ligne contenant /<chaîne>/.
Si /<chaîne>/ n'est pas trouvée :
1. Le message « String Not Found » (chaîne non trouvée) est affiché dans la zone commentaires de l'écran ÉDITEUR.
2. Si la recherche est effectuée avec la commande Z (ZONE) active, l'amplitude de la zone n'est pas modifiée.
FIND ne fait pas de différence entre les minuscules et les majuscules. Pour FIND « ABC » et « abc » sont des chaînes équivalentes. Le premier caractère après F, « / » dans l'exemple ci-dessus, est un délimiteur et peut être remplacé par tout autre caractère non alphabétique. F/<chaîne>/ est équivalent à F%(chaîne)%.

Exemple 1 : ZONE inactive

```
*********MOIRAGE**********
* Programme de moirage de la memoire
* point
        ORG     ENDMEM-$400 1K Reserve
DIRECT  EQU     *<-8    PAGE 0
        SETDP   DIRECT
        TITLE   Balayage Ecran
```

```
>F/DIR/
```

ENTREE

```
**********MOIRAGE**********
* Programme de moirage de la memoire
* point
        ORG     ENDMEN-$400 1K Reserve
DIRECT  EQU     *<-8    PAGE 0
        SETDP   DIRECT
        TITLE   Balayage Ecran
```

>F#ATE#

ENTREE

```
        ORG     ENDMEN-$400 1K Reserve
DIRECT  EQU     *<-8    Page 0
        SETDP   DIRECT
        TITLE   Balayage Ecran
*       INCLUD EQUATE       fichier con-
* tenant les Principaux codes d'entree
* du Moniteur MO5
```

Exemple 2 : **ZONE** active

>Z

ENTREE

>F&BIT&

ENTREE

```
*       INCLUD EQUATE       fichier con-
* tenant les Principaux codes d'entree
* du Moniteur MO5
        PAGE
START   LDA     PORTA   Mise en memoire
* Points Par mise a 1 du bit 0 du Port A
        ORA     #$1
```

>F % PAW %

ENTREE

>F%PAW%
String Not Found

REPLACE : Remplacement d'une chaîne de caractères par une autre chaîne dans le programme source résidant

Commande : Tapez R/<chaîne1>/<chaîne2>/ puis ENTREE
ou R puis ENTREE

Effet : La commande R/<chaîne1>/<chaîne2>/ (REPLACE) remplace, sur la ligne courante, <chaîne1> par <chaîne2>.
<chaîne1> et <chaîne2> peuvent être situées dans n'importe lequel des quatre champs de ligne de programme ou bien être incluses dans d'autres chaînes.

La commande REPLACE utilisée avec la commande ZONE active, remplace toutes les <chaîne1> par les <chaîne2> dans l'espace défini par ZONE.

<chaîne1> ne différencie pas les caractères minuscules et majuscules.
Pour <chaîne1> : <rrr> et <RRR> sont des chaînes équivalentes.
<chaîne2> remplace <chaîne1> en tenant compte des caractères minuscules et majuscules de <chaîne2> dans la commande.
L'écriture en minuscules d'un caractère sur la ligne de commande est obtenue avec le clavier en fonction minuscule, soit avec la touche jaune appuyée en même temps que le caractère souhaité.

Après le ou les remplacements de <chaîne1> par <chaîne2> le curseur reprend sa position initiale.
Si <chaîne1> et <chaîne2> sont omises dans la commande ce sont les dernières chaînes entrées avec la commande R (REPLACE) qui seront utilisées. Si <chaîne1> n'est pas trouvée sur la ligne courante ou dans la zone sélectionnée, le message « String Not Found » (chaîne non trouvée) est affiché dans la zone commentaire de l'écran ÉDITEUR.

Notez que la commande REPLACE ne modifie pas l'amplitude de la zone sélectionnée.
Comme pour FIND, le délimiteur « / » peut être remplacé par tout autre caractère non alphabétique.
R/<chaîne1>/<chaîne2>/ est équivalent à R&<chaîne1>& <chaîne2>&.

Exemple-1 : **ZONE** inactive

```
* alternance de 1 et 0 sur la ligne
M0      STD     ,X++    Charger l'ecran
        LEAY    -1,Y
```

```
>R/M0/IC/
```

ENTREE

```
* alternance de 1 et 0 sur la ligne
IC      STD     ,X++    Charger l'ecran
        LEAY    -1,Y
```

↓

```
IC      STD     ,X++    Charger l'ecran
        LEAY    -1,Y
        BNE     M0      Repete 20 fois
```

```
>R
```

ENTREE

Ecran inchangé

```
>R
String Not Found
```

↓

```
>R
```

ENTREE

```
IC      STD     ,X++    Charger l'ecran
        LEAY    -1,Y
        BNE     IC      Repete 20 fois
```

Exemple 2 : **ZONE** active

```
>Z
```

ENTREE

```
M2      LDY     #20     Compteur colonne:
* On affiche 20 fois 2 octets,soit
* 40 octets par ligne
        LDD     #$5555  Moirage:
```

```
>R/OCteT/Mot-8/
```

ENTREE

```
M2      LDY     #20     Compteur colonne:
* On affiche 20 fois 2 Mot-8s,soit
* 40 Mot-8s par ligne
        LDD     #$5555  Moirage:
```

PRINT : Impression sur l'imprimante du programme source résidant

Commande : Tapez P puis ENTREE

Le système répond : "Printer Ready Y/N?"

Effet : La commande P(PRINT) permet la copie sur l'imprimante de :
— La zone de programme source sélectionnée si la commande ZONE est active.
— La totalité du programme source résidant si la commande ZONE est inactive.
Avant l'impression du programme sur l'imprimante, le système pose la question :
"Printer Ready Y/N?" (l'imprimante est-elle prête ?)
Une réponse positive active l'imprimante.

Exemple :

```
>P
```

ENTREE

```
>P
Printer Ready Y/N?
```

D

L'imprimante est activée.
Après l'impression, la fin de la page est éjectée et l'imprimante se positionne au sommet de la page suivante.

NEW : Effacement du contenu de l'éditeur

Commande : Tapez N puis ENTREE

Le système répond : "Are you sure Y/N?"

Effet : La commande N (NEW) efface le programme source résidant dans l'éditeur.
Avant l'exécution de cette commande le système pose la question "Are you sure Y/N?" (êtes-vous sûr O/N ?). Si la réponse est positive la commande NEW est exécutée. L'écran est effacé ainsi que l'éventuel descripteur de fichier dans la zone de commande.
Notez que la « mémoire de travail » n'est pas effacée par la commande NEW.

Exemple :

```
>N
```

ENTREE

```
>N
Are you sure Y/N?
```

Y

L'écran et le contenu de l'éditeur sont effacés.

EXIT : Passage sous contrôle du moniteur

Commande : Tapez **X** puis **ENTREE**

Effet : La commande **X** (**EXIT**) permet au système de quitter l'éditeur et de passer sous contrôle du moniteur.
A l'issue de la commande **EXIT** le systeme génère l'écran du moniteur et attend une commande moniteur.
Le contenu de l'éditeur est préservé après une commande **EXIT**.
La commande **EXIT** permet également de charger un fichier binaire en passant sous contrôle du moniteur.

Exemple :

```
>X
```

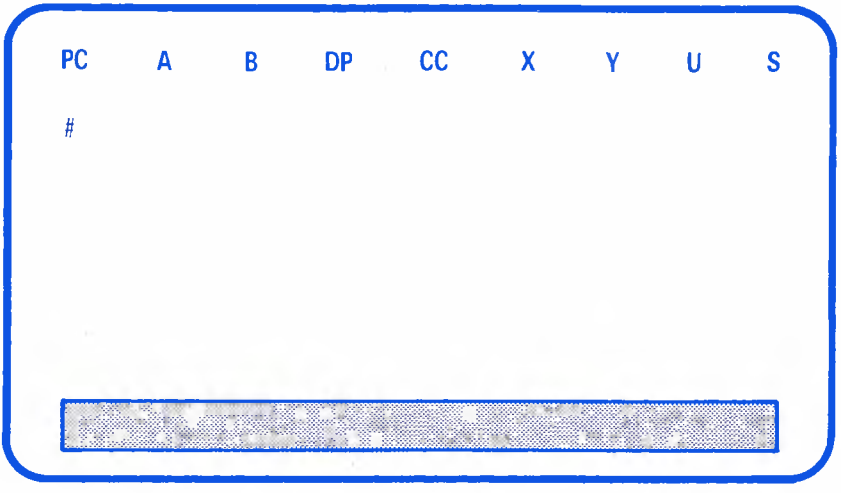
ENTREE

ou

```
>X<descripteur>
```

ENTREE

L'écran du moniteur est affiché.



Note : le retour sous contrôle de l'éditeur peut être obtenu en tapant **X** puis **ENTREE**.

QUIT : Retour sous contrôle de MENU

Commande : Tapez Q puis ENTREE

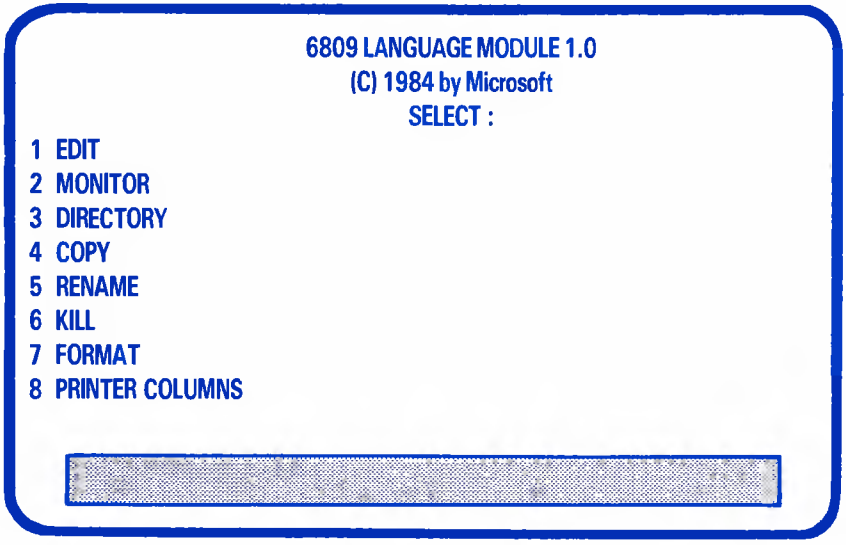
Effet : La commande Q (QUIT) redonne la main au MENU de la cartouche éditeur assembleur.
A l'issue de la commande QUIT le système génère MENU et attend une commande.
Le contenu de l'éditeur est préservé après une commande QUIT.

Exemple :

```
>Q
```

ENTREE

MENU est affiché



Note : le retour sous contrôle de l'éditeur peut être obtenu en tapant 1 puis ENTREE

2.3 Les commandes de gestion de fichiers sous ÉDITEUR

Les commandes de gestion de fichiers sont activables en mode COMMANDE ÉDITEUR. Elles permettent la gestion de fichiers contenant des programmes source.
Les commandes de fichiers peuvent être un caractère ou bien un caractère suivi d'un descripteur de fichier.
Si la commande est un caractère seul, le descripteur éventuellement présent dans la partie gauche de la ligne de commande sera utilisé pour compléter la commande.
Le suffixe .ASM et les périphériques C: ou Ø: peuvent être pris par défaut par le système.
Les commandes de gestion de fichier sous ÉDITEUR sont :
LOAD - SAVE - MERGE - VERIFY - EOF

LOAD : Chargement dans l'éditeur d'un programme source

Commande : Tapez **L** <descripteur de fichier> ENTREE
ou **L** ENTREE

Effet : La commande **L** <descripteur> **(LOAD)** charge un programme source à partir du périphérique spécifié dans le descripteur de fichier. Le chargement effectué, le descripteur de fichier qui est pris en compte par la commande **LOAD** est affiché dans la partie gauche de la zone de commande.
Pour utiliser la commande **LOAD**, un descripteur de fichier n'est pas toujours obligatoire :
— Avec l'emploi d'un LEP, c'est le premier fichier trouvé ayant .ASM pour suffixe qui sera chargé.
— Si un descripteur de fichier est présent dans la partie gauche de la ligne de commande, il peut être utilisé dans la commande **LOAD**.
Si vous utilisez un LEP, le système pose la question : "Cassette Ready Y/N?" (LEP prêt O/N ?). Si la réponse est positive, les messages "Searching" (recherche), "Skip" (saute) et "Found" (trouvé) sont affichés. Le message "Skip <nom de fichier>" est affiché sur la ligne commentaires, pendant la phase de recherche, chaque fois qu'un fichier est sauté.

Exemple 1 : Avec un lecteur de disquettes

```
                    >LØ : MOIRAGE
```

ENTREE

```
Ø : MOIRAGE.ASM     >
```

Le suffixe .ASM est pris par défaut.
ou :

```
                    >L MOIRAGE
```

ENTREE

```
Ø : MOIRAGE.ASM     >
```

Le lecteur Ø est pris par défaut.
Ou :

```
Ø : MOIRAGE.ASM     >L
```

ENTREE

```
Ø : MOIRAGE.ASM     >
```

Le descripteur de gauche est utilisé dans la commande

Exemple 2 : Avec un lecteur enregistreur de programmes (LEP)

```
                    >LC : MOIRAGE
```

ENTREE

```
                    >LC : MOIRAGE
Cassette Ready Y/N?
```

Y

```
                    >LC : MOIRAGE
Searching
```

```
                    >LC : MOIRAGE
Skip < nom de fichier >
```

```
0 : MOIRAGE.ASM     >
Found : MOIRAGE .ASM
```

SAVE : Sauvegarde le programme source résidant dans l'éditeur

Commande : Tapez **S** <descripteur de fichier> ENTREE
ou **S** ENTREE

Effet : La commande **S**<descripteur>**(SAVE)** sauvegarde, sur le périphérique spécifié dans le descripteur de fichier, la partie de programme sélectionnée par **ZONE**. Si **ZONE** est inactive tout le programme source en mémoire est sauvegardé.
La sauvegarde effectuée, le descripteur de fichier qui est pris en compte par la commande **SAVE** est affiché dans la partie gauche de la zone de commande.

1. Le suffixe .ASM est pris par défaut.
2. Si un autre suffixe est souhaité il doit être précisé dans la commande.
3. Si un descripteur de fichier est présent dans la partie gauche de la ligne de commande, il peut être utilisé dans la commande **SAVE**. Si vous utilisez un LEP, le système pose la question : « Cassette Ready Y/N? » (LEP prêt O/N?) (ex.2). Si la réponse est positive le programme est sauvegardé.

Exemple 1 : avec un lecteur de disquettes

```
                    > SMOIRAGE
ENTREE
Ø:MOIRAGE.ASM       >
```

Exemple 2 : avec un lecteur enregistreur de programmes (LEP)

```
                    >SC : MOIRAGE.TO7
ENTREE
                    >SC : MOIRAGE.TO7
                    Cassette Ready Y/N?
Y
C : MOIRAGE.TO7     >
```

Exemple 3 : avec un LEP, en utilisant le descripteur de fichier défini auparavant.

```
C : MOIRAGE.TO7     >S
ENTREE
C : MOIRAGE.TO7     >
```

MERGE : Charge et fusionne un fichier avec le programme source résidant

Commande : Tapez M <descripteur de fichier> ENTREE
ou M ENTREE

Effet : La commande M <descripteur> (MERGE) charge le programme spécifié dans le descripteur de fichier et le fusionne, à partir de la ligne précédant la ligne courante, avec le programme source résidant dans l'éditeur. MERGE peut également, comme SAVE, utiliser le nom du fichier qui est dans la partie gauche de la ligne de commande. Le nom du fichier précédent n'est pas modifié par la commande MERGE. Le curseur reprend sa position initiale après la commande MERGE :

Exemple : Fusion des programmes « EQUATE » et « MOIRAGE »

```
DIRECT EQU    *<-8     Page 0
       SETDP  DIRECT
       TITLE  Balayage Ecran
*                              fichier con-
* tenant les principaux codes d'entree
* du Moniteur M05
```

```
Ø : MOIRAGE .ASM        >MEQUATE
```

ENTREE

```
       SETDP  DIRECT
       TITLE  Balayage Ecran
*                                fichier con-
*
* POINTS D'ENTREE DU MONITEUR M05
```

```
ESC    EQU    $1B      ESCAPE
CR     EQU    $D       CARRIAGE RETURN
* tenant les principaux codes d'entree
* du Moniteur M05
```

```
Ø :MOIRAGE .ASM         >
```

VERIFY : Vérification du contenu de l'ÉDITEUR

Commande : Tapez V <descripteur de fichier> ENTREE
ou V ENTREE

Effet : La commande V <descripteur> (VERIFY) compare le contenu du fichier défini dans le descripteur avec le contenu de l'éditeur.
Si les contenus sont semblables, le curseur se repositionne et attend une nouvelle commande. Si les contenus de l'éditeur et du fichier sont dissemblables, le message « Verification Error » (erreur de vérification) est envoyé sur la ligne de commentaires.
Les fichiers comparés doivent avoir obligatoirement la même longueur.
Notez que la comparaison peut être effectuée entre un fichier représentant la sauvegarde de la zone sélectionnée dans l'éditeur et la zone elle-même.

Exemple 1 : pas d'erreur

```
Ø : MOIRAGE .ASM        >V
```

ENTREE

```
Ø : MOIRAGE .ASM        >
```

Exemple 2 : avec erreur

```
        LDD     #$5555  Moirage:
* alternance de 0 et 1 sur la ligne
                (par exemple)
M@      STD     ,X++    Charger l'ecran
```

```
Ø : MOIRAGE .ASM        >V
```

ENTREE

```
Ø : MOIRAGE .ASM        >
Verification Error
```

EOF : Positionnement du Lecteur Enregistreur de Programmes

Commande : Tapez E <descripteur de fichier> ENTREE
ou E ENTREE

Effet : La commande E<descripteur> (END OF FILE) est utilisée avec le Lecteur Enregistreur de Programmes (LEP).
Elle permet de positionner le LEP APRES le fichier spécifié.
Après la commande le système pose la question : « Cassette Ready Y/N? » (LEP prêt O/N ?). Si la réponse est positive la commande est validée. Si aucun fichier n'est indiqué dans la commande, le LEP se positionne APRÈS le premier fichier rencontré.
Si aucun fichier n'est enregistré la bande défile jusqu'à la fin.
Notez que C : est pris par défaut dans le descripteur de la commande.

Exemple :

```
>EMOIRAGE
```

ENTREE

```
>EMOIRAGE
Cassette Ready Y/N?
```

Y

```
>EMOIRAGE
Searching
```

```
>EMOIRAGE
Skip <nom de fichier>
```

```
>
Found : MOIRAGE .ASM
```

CHAPITRE 3
ASSEMBLAGE

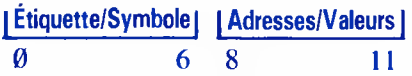
Après avoir écrit un programme source il convient de l'assembler. L'assemblage est une étape importante dans la phase de développement d'un programme. La tâche de l'assembleur sera de traduire le programme source en langage machine (programme objet), de construire la table des symboles utilisée par le programme, d'indiquer les erreurs rencontrées en cours d'assemblage et de les comptabiliser.
L'assembleur ne peut détecter que :
— des erreurs syntaxiques qui sont des erreurs liées aux règles d'écriture (des fautes d'orthographe dans le langage 68Ø9 en quelque sorte). Par exemple : LBD à la place de LDB ;
— des erreurs de non validité d'un opérateur, d'une étiquette etc. Par exemple : branchement à une étiquette inconnue. Un programme assemblé sans erreur ne garantit pas que le programme puisse s'exécuter correctement. Dans ce cas c'est la logique du programme qui est en cause.
Avant d'assembler, l'utilisateur devra s'assurer que le programme source est chargé en mémoire, car l'assembleur ne peut traiter que des programmes source résidant dans l'éditeur.
Les OPTIONS D'ASSEMBLAGE permettent à l'utilisateur de contrôler les conditions d'assemblage.
Ces options sont :

/WE	Wait on Errors	(attente si erreur)
/NL	No Listing	(pas de listing objet)
/LP	Line Printer	(sortie sur imprimante)
/NO	No Object	(pas de programme objet en mémoire)
/NS	No Symbol	(pas de table des symboles)
/SS	Small Screen	(écran étroit)

Les OPTIONS d'ASSEMBLAGE sont cumulables dans la commande **ASM** (assemblage) A . Exemple : **A/LP/WE/SS**

En l'absence d'option, la répartition sur l'écran des programmes source et objet est la suivante :

Le programme source est affiché en bleu et le programme objet en jaune. Les messages d'erreur sont visualisés, en rouge, à partir de la colonne Ø, avant la ligne source qui a généré l'erreur.
Le compteur d'erreurs est affiché en bleu à la suite du listing des programmes source et objet.
La table des symboles (en bleu) est la dernière sur un listing de sortie d'assemblage. Elle est située entre les colonnes Ø et 11.



Les DIRECTIVES d'ASSEMBLAGE sont des instructions réservées à l'assembleur et ne sont pas traduites en langage machine. Les directives d'assemblage permettent :
— l'affectation de symboles et d'étiquettes à des valeurs numériques. Ce sont les directives **EQU** et **SET** ;
— la gestion de la mémoire et l'implantation du programme objet. Ce sont les directives **RMB** et **ORG** ;
— la définition de constantes (huit et seize bits) et de textes codés en caractères ASCII. Ce sont les directives **FCB, FDB, FCC** ;
— la modification de l'adressage étendu en adressage direct. C'est la directive **SETDP** ;
— d'agir sur la sortie du listing d'assemblage. Ce sont les directives **PAGE** et **TITLE.**
— de définir la partie du programme à assembler et les programmes qu'il convient d'inclure dans la phase d'assemblage. Ce sont les directives **END** et **INCLUD.**
— d'appeler un sous-programme du moniteur MO5. Ce sont les directives **CALL, GOTO. STOP** permet de repasser sous contrôle du moniteur (debugger).

L'assemblage terminé, les DIRECTIVES D'ASSEMBLAGE **ne jouent plus aucun rôle.**

ATTENTION ! La commande d'assemblage d'un programme source est une commande ÉDITEUR, accessible en mode COMMANDE ÉDITEUR.

* Bleu tramé dans le manuel.

ASSEMBLAGE : Assemblage du programme source résidant dans l'éditeur

Commande : Tapez A<descripteur de fichier>/<option> ENTREE
ou A<descripteur de fichier> ENTREE
ou A/<Option> ENTREE
ou A ENTREE

Effet : La commande A <descripteur>/<option> (ASSEMBLAGE) est la commande d'assemblage du programme source résidant dans l'éditeur.

Si un descripteur de fichier est présent dans la commande d'assemblage, il spécifie la destination (disquette ou cassette) et le nom du programme objet, **sinon, le programme est assemblé en mémoire.** Si la partie gauche de la ligne de commande éditeur contient le nom du programme source résidant, le descripteur de fichier dans la commande d'assemblage peut être réduit au périphérique par défaut : « Ø : » ou « C : ». Dans ce cas le nom du fichier source sera utilisé avec le suffixe .BIN.

Si dans la commande d'assemblage le descripteur de fichier est omis, le programme objet sera assemblé en mémoire, à partir d'une adresse qui dépend de la présence ou non de la directive d'assemblage ORG dans le programme source.

— Si ORG est présent, il précise l'adresse d'implantation.

— Si ORG n'est pas présent, le programme objet sera implanté en mémoire à partir de la première page libre en mémoire (après la table des symboles).

Notez qu'il est possible d'arrêter le déroulement du listing d'assemblage sur l'écran par la touche STOP. La reprise est obtenue en appuyant sur une touche quelconque.

Exemple d'assemblage en mémoire :

```
Ø:MOIRAGE .ASM        >A
```

ENTREE

Note : par souci de clarté les listings d'assemblage qui suivent sont présentés sur 8Ø colonnes.

```
                        **********MOIRAGE**********
                        * Programme de moirage de la memoire
                        * Point
9C00                            ORG     ENDMEM-$400 1K Reserve
            009C        DIRECT  EQU     *-8     Page 0
            9C                  SETDP   DIRECT
                                TITLE   Balayage Ecran
                                INCLUD EQUATE       fichier con-
                        *
                        * POINTS D'ENTREE DU MONITEUR MO5
                        *
            0000        RETMON  EQU     $00     Main moniteur
            0002        PUTCH   EQU     $02     Affichage
            000A        GETCH   EQU     $0A     clavier
            000C        KTST    EQU     $0C     Test clavier
            000E        DRAW    EQU     $0E     Ligne
            0010        PLOT    EQU     $10     Point
            0012        CARAC   EQU     $12     Caractere
            0024        RSCONT  EQU     $24     RS-232
            0020        K7CONT  EQU     $20     Lecteur de k7
            0018        GETLP   EQU     $18     Crayon optique
            0016        LPINT   EQU     $16     Interrupteur.
            001E        NOTE    EQU     $1E     Musique
            0014        GETPT   EQU     $14     Lecture point
            001A        GETSC   EQU     $1A     Lecture caractere
            001C        JOYSTK  EQU     $1C     Manche a balais
            0026        DKCONT  EQU     $26     Controleur disque
            0004        MIMECO  EQU     $04     Memoire couleur
            0006        MIMEFO  EQU     $06     Memoire forme
            0008        BIP     EQU     $08     Emission d'un BIP
            0022        MOTLEP  EQU     $22     M/A moteur LEP
                        *
                        * Adresses Physiques
                        *
            A7C0        PORTA   EQU     $A7C0   Le bit 0 du Port A
                        * controle l'acces a la memoire ecran
                        * 1=Points ; 0=couleur
            0000        STADR   EQU     $0000   Debut de l'ecran
            1F40        ENDADR  EQU     $1F40   Fin de l'ecran +1
                        *
                        * CODES ASCII
                        *
            001F        US      EQU     $1F     UNIT SEPARATOR
            000C        FF      EQU     $C      FORM FEED
            0004        EOT     EQU     4       END OF TRANSM
            001B        ESC     EQU     $1B     ESCAPE
            000D        CR      EQU     $D      CARRIAGE RETURN
                        * tenant les Principaux codes d'entree
                        * du Moniteur MO5
                            PAGE
9C00 B6     A7C0        START   LDA     PORTA   Mise en memoire
                        * Points:Par mise a 1 du bit 0 du Port A
9C03 8A     01                  ORA     #$1
9C05 B7     A7C0                STA     PORTA
9C08 8E     0000                LDX     #STADR  Adresse de debut
                        * de l'ecran
```

```
9C0B 108E 0014    M2      LDY     #20       Compteur colonne
                  * on affiche 20 fois 2 octets,soit
                  * 40 octets par ligne
9C0F CC   5555            LDD     #$5555    Moirage
                  * alternance de 1 et 0 sur la ligne
9C12 ED   81      M0      STD     ,X++      Charger l'ecran
9C14 31   3F              LEAY    -1,Y
9C16 26   FA              BNE     M0        Repete 20 fois
9C18 108E 0014            LDY     #20       Ligne suivante
9C1C CC   AAAA            LDD     #$AAAA    Le motif est inver
                  * se pour decaler les 1 et les 0 d'une
                  * ligne a l'autre et obtenir un moirage
9C1F ED   81      M1      STD     ,X++
9C21 31   3F              LEAY    -1,Y
9C23 26   FA              BNE     M1        Toujours 20 fois
9C25 8C   1F40            CMPX    #ENDADR   Fin d'ecran
9C28 23   E1              BLS     M2        Sinon on recom
                  * mence 2 lignes a motifs alternes
9C2A BD   B000            STOP              Retour au moniteur
          9C00            END     START

00000 Total Errors
```

PROGRAMME OBJET

COMPTEUR D'ERREURS

PROGRAMME SOURCE

TABLE DES SYMBOLES

```
BIP     0008        KTST    000C
CARAC   0012        LPINT   0016
CR      000D        M0      9C12
DIRECT  009C        M1      9C1F
DKCONT  0026        M2      9C0B
DRAW    000E        MIMECO  0004
ENDADR  1F40        MIMEFO  0006
EOT     0004        MOTLEP  0022
ESC     001B        NOTE    001E
FF      000C        PLOT    0010
GETCH   000A        PORTA   A7C0
GETLP   0018        PUTCH   0002
GETPT   0014        RETMON  0000
GETSC   001A        RSCONT  0024
JOYSTK  001C        STADR   0000
K7CONT  0020        START   9C00
                    US      001F
```

Note : la table des symboles est sur une colonne à l'écran.

3.1 Options d'assemblage

/WE : Arrête l'assemblage si une erreur est détectée

Commande : Tapez **A <descripteur de fichier>/WE** ENTREE
ou **A/WE** ENTREE

Effet : À chaque erreur détectée, l'option d'assemblage **/WE** affiche le message correspondant à l'erreur et arrête l'assemblage en cours. À ce niveau l'utilisateur dispose de trois possibilités pour reprendre la main. En tapant :

— **CNT/C**, le système abandonne l'assemblage et retourne sous contrôle de l'éditeur

— **C**, (minuscule ou majuscule) le système continue l'assemblage mais ne s'arrêtera pas sur les erreurs potentielles dans la suite du programme.

— sur une touche quelconque le système reprend l'assemblage et s'arrêtera sur la prochaine erreur détectée.

Exemple : Écrivons une erreur dans le programme « MOIRAGE », par exemple :

```
STOP                    Retour au moniteur
NED      START
```

```
Ø : MOIRAGE .ASM        >A/WE
```

ENTREE

```
BC2A BD    B000          STOP              Retour au moniteur
Bad OPcode
        NED     START
Missing END Statement

00002 Total Errors
```

/NL : Supprime l'affichage du listing

Commande : Tapez A <descripteur de fichier>/NL ENTREE
ou A/NL ENTREE

Effet : L'option d'assemblage /NL supprime l'affichage du listing du programme.

- Restent seulement visualisés :

— Les messages d'erreur
— Le compteur d'erreurs
— La table des symboles

Exemple :

```
Ø : MOIRAGE .ASM        >A/NL
```

ENTREE

```
00000 Total Errors
BIP      0008
CARAC    0012
CR       000D
DIRECT   009C
DKCONT   0026
DRAW     000E
ENDADR   1F40
EOT      0004

RETCAR   0000
RSCONT   0024
STADR    0000
START    9C00
US       001F
```

/LP : Commutation de l'impression sur l'imprimante.

Commande : Tapez A <descripteur de fichier> /LP ENTREE
ou A/LP ENTREE

Effet : L'option d'assemblage /LP commute l'impression des listings de sortie d'assemblage sur l'imprimante.
Après l'impression, la fin de la page est éjectée et l'imprimante se positionne en haut de la page suivante.

Exemple :

```
Ø : MOIRAGE .ASM        > A/LP
```

ENTREE

L'imprimante est activée.

/NO : Suppression du programme objet généré en mémoire

Commande : Tapez A/NO ENTREE

Effet : L'option d'assemblage /NO supprime le programme objet généré en mémoire. Si un descripteur de fichier et l'option /NO sont spécifiés en même temps dans la commande d'assemblage, l'option /NO sera ignorée, car de toutes façons le descripteur de fichier dirige l'assemblage vers un périphérique. Le programme objet sera sauvegardé dans le fichier spécifié.

Exemple :

```
Ø : MOIRAGE .ASM        > A/NO
```

ENTREE

L'assemblage est effectué, mais le programme objet n'est pas implanté en mémoire. Cela sert principalement à vérifier l'absence d'erreur de syntaxe.

/NS : Suppression de la table des symboles

Commande : Tapez A <descripteur de fichier> /NS ENTREE
ou A/NS ENTREE

Effet : L'option d'assemblage /NS supprime la table des symboles dans le listing de sortie d'assemblage.

Exemple :

```
Ø : MOIRAGE .ASM        > A/NS
```

ENTREE

Le listing est visualisé à l'écran sans la table des symboles.

/SS : Édition sur des lignes séparées des programmes source et objet

Commande : Tapez A <descripteur de fichier> /SS ENTREE
ou A/SS ENTREE

Effet : L'option d'assemblage /SS édite les programmes source et objet sur des lignes différentes.

Exemple :

```
Ø : MOIRAGE .ASM        > A/SS
```

ENTREE

```
* mence 2 lignes a motifs alternes
9C2A BD   B000
          STOP              Retour au moniteur
             9C00
          END     START

00000 Total Errors
```

3.2 Directives d'assemblage

Directive EQU : EQUATE (affectation d'une valeur à un symbole)

Syntaxe : <symbole> EQU <expression> <commentaire>

But : La directive d'assemblage **EQU** assigne au symbole placé dans le champ étiquette la valeur de l'expression (8 ou 16 bits) placée dans le champ opérande. Ces équivalences sont consignées dans la table des symboles.

Remarques : **EQU** est une directive qui donne au symbole une valeur qui n'est pas liée au compteur de programme (PC).
Les symboles définis avec **EQU** ne peuvent pas être redéfinis dans la suite du programme.
Les symboles utilisés dans la définition d'autres symboles ou le calcul d'une expression doivent être définis au préalable.
Pour faciliter la compréhension du programme, il est pratique de placer les directives **EQU** au début du programme.

Exemple :
Affectation d'une valeur à un symbole.

Prog.objet	Prog. source			
0008	BIP	EQU	$08	Emission d'un BIP
1F40	ENDADR	EQU	$1F40	Fin de l'ecran +1
001F	US	EQU	$1F	UNIT SEPARATOR

Affectation d'un symbole à un autre symbole.

000E	DRAW	EQU	$0E	Ligne
000E	TRAIT	EQU	DRAW	TRAIT=DRAW=$08

Affectation d'un symbole à une expression.

0010	PLOT	EQU	$10	Point
0018	GETLP	EQU	PLOT+BIP	GETLP=$10+$08=$18

Directive SET : Set symbol to value (affectation temporaire d'une valeur à un symbole)

Syntaxe : <symbole> **SET** <expression> <commentaire>

But : La directive d'assemblage **SET** assigne au symbole placé dans le champ étiquette la valeur de l'expression (8 ou 16 bits) placée dans le champ opérande. Ces équivalences sont consignées dans la table des symboles et repérés par le caractère S (comme SET).

Remarques : **SET** est une directive d'assignation semblable à **EQU** mais contrairement à **EQU**, les symboles définis avec **SET** peuvent être redéfinis dans la suite du programme, avec une directive **SET**.
Les symboles utilisés dans la définition d'autres symboles ou le calcul d'une expression doivent être définis au préalable.
La directive **SET** est très utile pour définir temporairement des symboles ou des étiquettes réutilisables dans la suite du programme.
Note : La table des symboles indique la dernière assignation des symboles définie par la directive **SET**.

Exemple : La séquence ci-dessous montre l'emploi de la directive **SET**.

Prog. objet	Prog. source		
4532	CHIFF	SET	$4532
3678	CHIFF	SET	$3678
0004	VAL	SET	$04
0009	VAL	SET	VAL+VAL
0040	VAL	SET	VAL*VAL
0000		END	

```
00000 Total Errors
CHIFF   3678     S
VAL     0040     S
```

Directive RMB : Reserve Memory Bytes (réservation d'octets en mémoire)

Syntaxe : <symbole> RMB <expression> <commentaire>

But : La directive d'assemblage **RMB** réserve une zone en mémoire définie par l'expression contenue dans le champ opérande. Pour cela **RMB** ajoute au compteur de programme (PC) la valeur de l'expression.

Remarques : **RMB** réserve un espace mémoire non initialisé à une valeur particulière. Les symboles utilisés dans la définition ou le calcul de l'expression du champ opérande doivent être définis avant la rencontre de la directive **RMB**.
Avec la directive **RMB**, le symbole présent dans le champ étiquette est assigné à la valeur courante du compteur de programme.
Dans le cas d'un assemblage avec un <descripteur de fichier> aucun code n'est créé dans le fichier binaire par la directive **RMB**. Ainsi, le fichier .BIN ne pourra pas se comparer avec l'image mémoire chargée par **VERIFY** dans le moniteur.

Exemple : La séquence ci-dessous montre l'emploi de la directive **RMB**.

Prog. objet

```
6900                        ORG    $6900
6900                        RMB    16
6910              RES1      RMB    16
6920              RES2      RMB    16*2
          6940    FIN       EQU    *
          0000              END

00000 Total Errors
FIN     6940
RES1    6910
RES2    6920
```

Prog. source

```
6900 +----------+ DEBUT OBJET
     | 16 octets |
6910 +----------+ RES1
     | 16 octets |
6920 +----------+ RES2
     | 32 octets |
6940 +----------+ FIN
```

Directive ORG : Origine (initialisation du compteur de programme)

Syntaxe : ORG <expression> <commentaire>

But : La directive d'assemblage ORG charge le compteur de programme (PC) à la valeur spécifiée par l'expression contenue dans le champ opérande. Les instructions qui suivent ORG sont assemblées en incrémentant PC. ORG définit ainsi le début du programme objet qui résulte de l'assemblage.

Remarques : Il peut y avoir plusieurs ORG dans un programme source.
La directive ORG ne doit pas être écrite avec une étiquette.
Si ORG est omise dans le programme source, le compteur de programme est chargé à :
— ØØØØ si l'assemblage est demandé avec un fichier destination. Attention : un programme assemblé en ØØØØ ne se charge pas sur MO5 (sauf si un offset est précisé).
— la valeur de la première page qui suit la table des symboles, si l'assemblage est demandé en mémoire.
Une page de mémoire correspond aux 256 octets définis entre deux valeurs consécutives de poids forts d'adresse. Par exemple :

L'étiquette définissant la fin de la mémoire disponible (ENDMEM) peut être utilisée dans l'expression pour calculer le début du programme objet.

Exemple : Les séquences ci-dessous montrent les conséquences de la présence ou de l'absence de la directive ORG dans le programme source.
A- ORG est présent et l'expression utilise ENDMEM

```
Ø : MOIRAGE .ASM        >A
```

ENTREE

```
                        * Point
9C00                          ORG     ENDMEM-$400 1K Reserve
            009C        DIRECT EQU      *<-8      Page Ø
            9C                SETDP   DIRECT

9C00 B6     A7C0        START   LDA     PORTA     Mise en memoire
                        * Points Par mise a 1 du bit Ø du Port A
```

B- **ORG** est présent et l'expression contient l'adresse de début du programme objet.

```
Ø : MOIRAGE.ASM        >A
```

ENTREE

```
                    * Point
9000                      ORG     $9000
          0090      DIRECT EQU     *<-8     Page 0
          90                SETDP   DIRECT

9000 B6   A7C0      START  LDA     PORTA     Mise en memoire
                    * Points:Par mise a 1 du bit 0 du Port A
```

C- **ORG** est absent et la commande ne contient pas de fichier destinataire.

```
Ø : MOIRAGE.ASM        >A
```

ENTREE

```
                    * Point
          002E      DIRECT EQU     *<-8     Page 0
          2E                SETDP   DIRECT

2E00 B6   A7C0      START  LDA     PORTA     Mise en memoire
                    * Points:Par mise a 1 du bit 0 du Port A
```

D- **ORG** est absent et la commande précise le fichier destinataire.

```
Ø : MOIRAGE.ASM        >A0 :
```

ENTREE

```
                    * Point
          0000      DIRECT EQU     *<-8     Page 0
          00                SETDP   DIRECT

0000 B6   A7C0      START  LDA     PORTA     Mise en memoire
                    * Points:Par mise a 1 du bit 0 du Port A
```

Directive FCB : Form Constant Byte (définition d'une constante d'un octet)

Syntaxe : <symbole> FCB <expr>, <expr>, <expr> <commentaire>

But : La directive d'assemblage **FCB** stocke des constantes de 8 bits, dans des positions mémoire définies par la valeur courante du compteur de programme (PC). Pour chaque octet chargé, le PC est incrémenté de 1.

Remarques : **FCB** crée un espace mémoire initialisé à une valeur particulière.
FCB peut définir plusieurs constantes séparées par une virgule.
Les constantes peuvent être des valeurs numériques , des caractères, des symboles, des expressions.
Les constantes supérieures à huit bits seront tronquées à huit bits et signalées par le message « Operand Too Large » (opérande trop grand).
Les symboles utilisés dans la définition ou le calcul de l'expression contenue dans le champ opérande peuvent être définis après la rencontre de la directive **FCB**.
Avec la directive **FCB**, le symbole présent dans le champ étiquette est assigné à la valeur courante du compteur de programme.
Les constantes sont restituées sur le listing du programme objet à raison de quatre octets par ligne.

Exemple : La séquence ci-dessous montre l'emploi de la directive **FCB**.

Prog.objet	Prog. source		
6900		ORG	$6900
6900 04 08 41 60	VAL	FCB	$04,$08,'A,$20+$40
Operand Too Large			
6904 FF 00	VAL4	FCB	255,256
6906 34	VAL6	FCB	TRE
0034	TRE	EQU	$34
0000		END	

Directive FDB : Form Double Byte Constant (définition d'une constante de deux octets ou un mot)

Syntaxe : <symbole> FDB <expr>, <expr>, <expr> <commentaire>

But : La directive d'assemblage FDB stocke des constantes de seize bits, dans des positions mémoire définies par la valeur courante du compteur de programme (PC). Pour chaque mot chargé, le PC est incrémenté de 2.

Remarques : FDB crée un espace mémoire initialisé à une valeur particulière.
FDB peut définir plusieurs constantes séparées par une virgule.
Les constantes peuvent être des valeurs numériques, des caractères, des symboles, des expressions.
Les constantes supérieures à seize bits seront tronquées à seize bits.
Les symboles utilisés dans la définition ou le calcul de l'expression contenue dans le champ opérande peuvent être définis après la rencontre de la directive FDB.
Avec la directive FDB, le symbole présent dans le champ étiquette est assigné à la valeur courante du compteur de programme.
Les constantes sont restituées sur le listing du programme objet à raison de deux mots par ligne.

Exemple : La séquence ci-dessous montre l'emploi de la directive FDB.

Prog.objet	Prog. source		
6900		ORG	$6900
6A00 0408 0041 6A04 6000	MOT	FDB	$0408,'A,$2000+$4000
6A06 FFFF 0001	MOT6	FDB	65535,65537
6A0A 5678	MOT10	FDB	TRU
5678	TRU	EQU	$5678
0000		END	

Directive FCC : Form Constant Character String (définition d'une constante chaîne de caractères)

Syntaxe : <symbole> FCC <déli><chaîne><déli><commentaire>

But : La directive d'assemblage **FCC** stocke des chaînes de caractères dans des positions mémoires définies par la valeur courante du compteur de programme (PC). À chaque caractère le PC est incrémenté de 1.

Remarques : **FCC** crée un espace mémoire initialisé à une valeur particulière.
FCC définit la chaîne de caractères à stocker entre deux délimiteurs identiques.
Ces délimiteurs peuvent être n'importe lequel des caractères imprimables.
L'espace utilisé dans le champ opérande provoque une tabulation automatique. Pour cette raison il est préférable d'utiliser **CNT/S** (espace en vidéo inversée) pour générer un espace dans une chaîne de caractères.
Avec la directive **FCC**, le symbole présent dans le champ étiquette est assigné à la valeur courante du compteur de programme.
Les chaînes sont restituées sur le listing du programme objet à raison de quatre caractères par ligne.
Notez que **FCC /A/** est équivalent à **FCB 'A**

Exemple : La séquence ci-dessous montre l'emploi de la directive **FCC**.

Prog.objet	Prog. source		
6900		ORG	$6900
6900 54 52 4F 50	TG	FCC	!TROP GRAND!
6904 20 47 52 41			
6908 4E 44			
690A 41 55 20 44	TP	FCC	#AU DESSOUS DE VALEUR#
690E 45 53 53 4F			
6912 55 53 20 44			
6916 45 20 4C 41			
691A 20 56 41 4C			
691E 45 55 52			
6921 45 52 52 45	ER	FCC	<ERREUR<
6925 55 52			
0000		END	

Directive SETDP : Set Direct Page
(positionnement de la page directe)

Syntaxe : SETDP <expression> <commentaire>

But : La directive d'assemblage SETDP définit une « page Ø », dans l'espace mémoire, où l'adressage étendu sera modifié en adressage direct. Cela pour accélérer la vitesse d'exécution et diminuer la longueur du programme objet (deux ou trois octets au lieu de trois ou quatre octets).

Remarques : SETDP n'admet pas d'étiquette.
La directive SETDP ne modifie pas le registre de page Ø (DP) du microprocesseur 68Ø9.
L'utilisateur devra prévoir son chargement en conformité avec la dernière valeur de la « page Ø ». L'omission de chargement du registre (DP) conduirait à une erreur à l'exécution du programme mais pas à l'assemblage. Le système ne peut connaître les intentions du programmeur !! La valeur du registre de page (DP) est appelée « page de base ».

A l'initialisation du système, la « page Ø » est positionnée à ØØ.
La directive SETDP peut être employée plusieurs fois dans une séquence de programme.

Le champ opérande d'une directive SETDP peut être occupé par : une constante, un symbole, une expression.
Les symboles utilisés dans la définition de SETDP doivent être définis en amont.
Si le calcul de l'expression de SETDP conduit à une valeur supérieure à huit bits, la « page Ø » sera chargée avec les huit bits de poids faible de l'expression et le message « Operand Too Large » (opérande trop grand) sera généré par l'assembleur.

Notez que l'utilisateur a toujours la possibilité de soustraire une instruction à l'adressage direct dans la « page Ø » en forçant l'assembleur à utiliser l'adressage étendu par (>).

Exemple : Les séquences ci-après montrent les diverses possibilités d'emploi de la directive SETDP.

Prog.objet		Prog. source
7000		ORG $7000
		* %%%%%%%%%%%%%%%%%%%%
		***SETDP N'EST PAS ENCORE PROGRAMME ***
		* %%%%%%%%%%%%%%%%%%%%
		* "PAGE 0":00
7000 0A	A3	DEC $00A3
7002 0C	44	INC $0044
	0070	DIRECT EQU *<-8
		* %%%%%%%%%%%%%%%%%%%%
		***SETDP UTILISE AVEC UNE CONSTANTE ***
		* %%%%%%%%%%%%%%%%%%%%
	80	SETDP $80 "PAGE 0":80
7004 0A	22	DEC $8022
7006 7C	9028	INC $9028
	90	SETDP $90 "PAGE 0":90
7009 7A	8022	DEC $8022
700C 8C	28	INC $9028
		* %%%%%%%%%%%%%%%%%%%%
		***SETDP UTILISE AVEC UNE EXPRESSION **
		* %%%%%%%%%%%%%%%%%%%%
OPerand Too Large		
	1A	SETDP $84+$96 "PAGE 0":XX1A
700E 0A	56	DEC $1A56
7010 0C	78	INC $1A78
		* %%%%%%%%%%%%%%%%%%%%
		FORCAGE DE L'ADRESSAGE ETENDU **
		* %%%%%%%%%%%%%%%%%%%%
7012 7A	1A56	DEC >$1A56
7015 7C	1A78	INC >$1A78
		* %%%%%%%%%%%%%%%%%%%%
		SETDP UTILISE AVEC UN SYMBOLE **
		* %%%%%%%%%%%%%%%%%%%%
	70	SETDP DIRECT "PAGE 0":80
7018 7A	B0AB	DEC $B0AB
701B 7C	B077	INC $B077
	0000	END

```
00001 Total Errors


  DIRECT  0070
```

Avertissement : SETDP est inactif quand l'adressage étendu est défini par un symbole spécifié en aval.

Directive PAGE : Saut de Page

Syntaxe : **PAGE**

But : La directive d'assemblage **PAGE** fait avancer le papier de l'imprimante au début de la page suivante.
Si l'impression n'est pas demandée (par **/LP**) la directive **PAGE** est sans effet.
Note : L'insertion de lignes blanches dans le listing se fait simplement en mettant des lignes blanches dans le programme source.

Directive TITLE : Titre (définition d'un titre)

Syntaxe : **TITLE** <chaîne>

But : La directive d'assemblage **TITLE** permet de spécifier un en-tête de page par la chaîne de caractères contenue dans le champ opérande. **TITLE** est active avec l'option d'assemblage **/LP**. Elle est sans effet dans les autres cas.

Remarques : La chaîne est délimitée par la fin de la ligne et peut contenir 26 caractères. Ce nombre peut être porté à 28, si le programmeur utilise le reste du champ code opération.
La chaîne spécifiée est imprimée en haut de chaque page tant qu'une autre directive **TITLE** n'a pas été rencontrée.
TITLE n'admet pas de symbole dans le champ étiquette, ni de commentaire.

Exemple :

```
TITLE  CECI EST UN TITRE*********

TITLEceci est encore un titre++++
```

Directive INCLUD : Inclusion d'un programme source

Syntaxe : **INCLUD** <descripteur> <commentaire>

But : La directive d'assemblage **INCLUD** fusionne, à l'assemblage, le fichier spécifié dans le champ opérande avec le programme source résidant en mémoire. C'est l'équivalent d'une commande M **(MERGE)** qui ne se réalise qu'au moment de l'assemblage.

Remarques : La directive **INCLUD** n'est pas utilisable avec un fichier résidant sur le lecteur de cassettes, pour les raisons suivantes :
— L'assembleur multipasse nécessite plusieurs passages sur le programme source. Il serait donc nécessaire de repositionner le lecteur plusieurs fois.
— Avec un lecteur de cassettes, il n'est pas possible, en même temps, de lire le programme source et de sauvegarder le programme objet.
INCLUD n'admet pas d'inclusion imbriquée : le programme à inclure ne doit donc pas contenir la directive **INCLUD**.
Si **INCLUD** est rencontrée pendant l'inclusion, le message d'erreur « BAD INCLUD » (erreur d'inclusion) est généré. Il en est de même si une tentative d'inclusion est faite à partir du LEP.

Exemple : La séquence ci-dessous montre l'emploi de la directive **INCLUD**.

Prog.objet	Prog. source

```
                TITLE  Balayage Ecran
*               INCLUD EQUATE      fichier con-
*
* POINTS D'ENTREE DU MONITEUR MO5
*
0000    RETCAR EQU     $00      Main cartouche ASM

001B    ESC    EQU     $1B      ESCAPE
000D    CR     EQU     $D       CARRIAGE RETURN
* tenant les Principaux codes d'entree
* du Moniteur MO5
               PAGE
```

Directive END : Fin du programme source

Syntaxe : END <expression> <commentaire>

But : La directive d'assemblage **END** marque la limite du programme source. Les lignes de programmes situées derrière **END** seront ignorées par l'assembleur.

Remarques : Si la directive d'assemblage **END** est omise à la fin d'un programme, le système générera le message d'erreur : « Missing **END** statement » (la directive **END** est manquante).
END n'admet pas de symbole dans le champ étiquette.
L'expression facultative contenue dans le champ opérande précise l'adresse d'exécution du programme. Cette valeur sera utilisée pour spécifier l'adresse d'exécution, dans le fichier, à la création du programme objet.

Exemple : la séquence ci-dessous montre l'emploi de la directive **END**.

```
START   LDA     PORTA   Mise en memoire
* Points:Par mise a 1 du bit 0 du Port A
        ORA     #$1

* mence 2 lignes a motifs alternes
        STOP            Retour au moniteur
        END     START
```

Directive CALL : Appel d'un sous-programme moniteur

Syntaxe : <**symbole**> **CALL** <**expression**> <**commentaire**>

But : La directive d'assemblage **CALL** permet d'appeler une routine du moniteur système. L'octet contenu dans le champ opérande spécifie le code du point d'entrée.

Remarques : **CALL** génère le code machine correspondant à un SWI suivi de l'octet contenu dans le champ opérande (voir annexe A, "accès aux routines du moniteur"). L'octet du champ opérande peut être défini par une expression. Si le calcul de l'expression conduit à une valeur supérieure à 8 bits, elle sera tronquée à 8 bits et signalée par le message "Operand Too Large".
Comme aucun test n'est effectué sur l'opérande, si la valeur de l'expression ne correspond par à un point d'entrée du moniteur, une anomalie de fonctionnement peut se produire et ainsi modifier des registres internes au système.
Les symboles utilisées dans l'expression peuvent être définis après la rencontre de la directive **CALL**. Les points d'arrêt sont permis sur **CALL** et l'exécution du programme peut être reprise par un CONTINUE. **CALL** est l'équivalent d'un JSR.

Exemple : La séquence ci-dessous montre l'emploi de la directive **CALL**.

```
6900                        ORG     $6900
6900 3F   04      COUL      CALL    $04
6902 3F   06      FORM      CALL    MIMEFO
6904 3F   08      SONNE     CALL    GETLP-PLOT    (08=BIP)
          0000                END

00000 Total Errors
```

Directive GOTO : Saut à un programme moniteur

Syntaxe : <**symbole**> **GOTO** <**expression**> <**commentaire**>

But : La directive d'assemblage **GOTO** permet d'aiguiller le programme principal vers le moniteur système. L'octet contenu dans le champ opérande spécifie le point d'entrée.

Remarques : **GOTO** génère le code machine correspondant à un SWI suivi de l'octet contenu dans le champ opérande et force le bit 7 à un (voir annexe A "accès aux routines du moniteur").
L'octet du champ opérande peut être défini par une expression. Si le calcul de l'expression conduit à une valeur supérieure à 8 bits, elle sera tronquée à 8 bits et signalée par le message "Opérande Too Large". Aucun test n'est effectué sur l'opérande. Si la valeur de l'expression ne correspond pas à un point d'entrée du moniteur, une anomalie de fonctionnement peut se produire et modifier les registres internes au système. Les symboles utilisés dans l'expression peuvent être définis après la rencontre de la directive **GOTO**. **GOTO** est l'équivalent d'un JMP. Les sous-programmes moniteur étant terminés par un RTS, à la rencontre de cette instruction le PC pointera sur l'instruction qui suit le dernier appel à un sous-programme. Les points d'arrêt sont permis sur un **GOTO**. L'exécution du programme peut être reprise par un **CONTINUE**.

Exemple : La séquence ci-dessous montre l'emploi de la directive **GOTO**.

```
6900                        ORG     $6900
6900 3F   84      COUL      GOTO    $04
6902 3F   84                GOTO    $84
6904 3F   86      FORM      GOTO    MIMEFO
6906 3F   88      SONNE     GOTO    GETLP-PLOT
          0000                END
```

Directive STOP : Retour en état "attente de commande moniteur"

Syntaxe : <**symbole**> **STOP** < > <**commentaire**>

But : La directive d'assemblage **STOP** arrête l'exécution du programme et redonne la main au moniteur qui se positionne dans l'état « attente de commande » « # ».

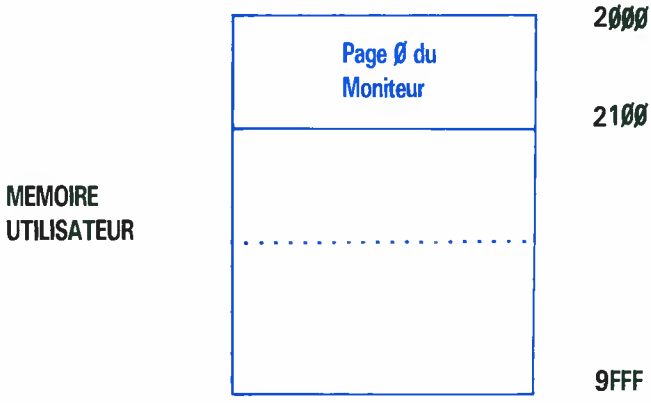
Remarques : **STOP** génère dans le programme objet l'instruction BD BØØØ (JSR $BØØØ) qui a pour effet de rendre la main au moniteur. **STOP** n'admet pas d'expression dans le champ opérande. La directive d'assemblage **STOP** n'admet pas de point d'arrêt. Cela n'aurait d'ailleurs aucun sens car un **STOP** génère déjà lui-même un point d'arrêt. Notez qu'il faut pas confondre **STOP** et **END**.
STOP est la directive de retour sous contrôle du moniteur tandis que **END** délimite la partie du programme source à assembler.

Exemple : La séquence ci-dessous montre l'emploi de la directive **STOP**.

```
9C2A BD   BØØØ   TRUC   STOP          Retour au moniteur
          9C00           END    START
```

Où est situé le programme objet ?
Quelle est la fin du programme source ?

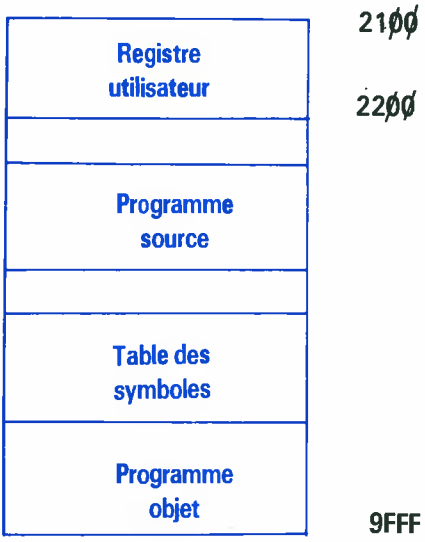
Voilà les questions que se posent bien des programmeurs, au début de l'utilisation d'un nouvel assembleur.
Avant de répondre à ces questions, il convient de rappeler l'emplacement de la mémoire utilisateur dans le champ adressable.



C'est dans cette zone de mémoire que doivent être logés :

- Le programme source.
- La table des symboles.
- Le programme objet.

La position du programme source est choisie par le système.
Il se situera au début de l'espace utilisateur.
Le programme source sera suivi de la table des symboles.
Le reste de la mémoire, que nous appelerons « mémoire libre », pourra être occupé par le programme objet.
La position du programme objet en « mémoire libre » est définie par la directive d'assemblage ORG (origine).
Sans directive d'assemblage ORG dans le programme source, le programme objet sera résidant à partir de la première page suivant la table des symboles.
Une étiquette spéciale ENDMEM définit la fin de la mémoire RAM.



ENDMEM peut être utilisée pour définir une position dans la mémoire.

Exemple : `ORG ENDMEM-$3000`

... est acceptable par l'assembleur

Note : « * » ou « . » sont des symboles spéciaux qui représentent la valeur courante du compteur de programme. Ces symboles peuvent être utilisés dans le calcul d'expressions.

Exemple : `DIRECT EQV */256 (division entiere)`

Une tentative d'implantation du programme objet (par ORG) dans la zone occupée par le programme source n'est pas autorisée et génère le message : « Bad memory » (mauvaise mémoire).

Il n'est donc pas possible d'écraser le programme source avec le programme objet.

Un programme source peut être suffisamment long pour ne pas laisser assez de place en mémoire pour l'implantation du programme objet.

Devant ce problème deux solutions sont envisageables :

1 - Demander l'assemblage du programme objet sur un fichier.

Dans ce cas toute la mémoire disponible est utilisée par le programme source.
2 - Sauvegarder le programme source sur un fichier et l'appeler seulement dans la phase d'assemblage par directive INCLUD.
Par exemple :

```
DEBUT  ORG     *
       INCLUD PROGSOU  (fichier source)
       END     DEBUT
```

Dans ce cas toute la mémoire (ou presque) est disponible pour implanter le programme objet.
Les commandes ci-dessous exposent, l'implantation du programme objet en fonction des paramètres d'entrée.

La directive d'assemblage ORG est présente dans le programme source.

Commande

Demande d'assemblage avec un descripteur de fichier.

>A <descripteur>

Résultat

Le programme objet est implanté sur le périphérique assigné. ORG spécifie l'adresse d'origine du programme objet.

Commande

Demande d'assemblage sans descripteur de fichier. Un fichier est présent dans la partie gauche.

<descripteur>	>AC

Résultat

Le programme objet est implanté sur le périphérique du fichier en cours. ORG spécifie l'adresse d'origine du programme objet.

Commande

Demande d'assemblage sans descripteur de fichier.

```
>A
```

Résultat

Le programme objet est implanté en mémoire à l'adresse spécifiée par ORG.

La directive d'assemblage ORG est absente du programme source.

Commande

Demande d'assemblage avec un descripteur de fichier.

```
>A <descripteur>
```

Résultat

Le programme objet est implanté sur le périphérique assigné avec l'adresse ØØØØ comme origine.*

Commande

Demande d'assemblage sans descripteur de fichier. Un fichier est présent dans la partie gauche.

```
<descripteur>        >AC
```

Résultat

Le programme objet est implanté sur le périphérique assigné et portera le nom du fichier en cours avec l'adresse ØØØØ comme origine.*

Commande

Demande d'assemblage sans descripteur de fichier.

```
>A
```

Résultat

Le programme objet est implanté en mémoire a partir de la première page qui suit la table des symboles.

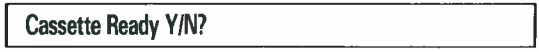
Note : Pour que les programmes * puissent être exécutés, il faut qu'ils soient relogeables sur MO5 .

CHAPITRE 4
LE MONITEUR

Le moniteur est une partie du programme résidant dans la mémoire de la cartouche ROM.
Le moniteur est à la fois un outil de vérification et de modification du contenu des mémoires et des registres internes du microprocesseur 68Ø9, et un outil d'exécution et de mise au point des programmes objet issus de l'assembleur.
Le moniteur offre la possibilité d'exécuter en totalité ou partiellement un programme , de connaître les valeurs intermédiaires des registres du 68Ø9 et le contenu des mémoires pour le modifier éventuellement.
Le moniteur utilise l'écran dans sa totalité et affiche en permanence, sur la ligne Ø, les noms des registres du microprocesseur.

La ligne 25 (rouge) est destinée à recevoir les commentaires et messages d'erreur du moniteur.

Exemple :



(cassette prête O/N ?)
Les entrées et sorties du moniteur apparaissent à l'écran :
— Les entrées en bleu sur fond noir
— Les sorties en jaune sur fond noir (bleu tramé dans le manuel)
L'état « Attente de commande moniteur » est matérialisé par « # » à gauche du curseur.

Accès au moniteur

Pour accéder au moniteur, l'utilisateur dispose de deux entrées :
— A partir de MENU.
— A partir de l'ÉDITEUR.

À partir de MENU :

La commande de MENU 2 active le bloc moniteur.
Sans descripteur de fichier :

2 MONITOR [ ] ENTREE

L'écran du moniteur est généré et positionné dans l'état d'attente commande moniteur

```
PC    A    B    DP    CC    X    Y    U    S

#
```

Avec descripteur de fichier :

2 MONITOR [Ø : MOIRAGE.BIN] ENTREE

ou 2 MONITOR [MOIRAGE.BIN] ENTREE

ou 2 MONITOR [MOIRAGE] ENTREE

ou 2 MONITOR [C : MOIRAGE] ENTREE

Note : Ø : C : et .BIN sont facultatifs et sont pris par défaut.
L'écran du moniteur est généré comme ci-dessus.
Le fichier objet spécifié dans la commande est chargé en mémoire aux adresses qui ont été définies au moment de la sauvegarde du programme objet.

À partir de l'ÉDITEUR :

La commande X en mode COMMANDE ÉDITEUR permet l'accès au moniteur (avec chargement éventuel d'un fichier objet).

[Ø : MOIRAGE .ASM >X<fichier>]

ENTREE

L'écran du moniteur est généré comme ci-dessus. Le passage de l'éditeur au moniteur ne modifie pas le contenu de la mémoire. Seuls les fichiers objet peuvent être chargés par le moniteur. Un appel de fichier source sous moniteur provoque l'erreur : « File format error » (erreur de format de fichier).

4.1 Commandes moniteur

Les commandes moniteur sont des commandes actives à partir du clavier, le moniteur étant en état « attente de commande ».

INPUT : entrée

Commande : Le moniteur en état « attente de commande » « # »
Tapez : I D ENTREE
ou : I H ENTREE

Effet : La commande I {H ou D} (INPUT) déclare le mode numérique d'entrée dans le moniteur.
Les codes admis sont :
DÉCIMAL : D
HEXADÉCIMAL : H
Le code non déclaré devra être précisé dans la commande par son préfixe ou son suffixe.
Une nouvelle déclaration annule la déclaration précédente.
Si la déclaration d'entrée est omise, le MONITEUR **interprétera les entrées en hexadécimal par défaut.**
Les valeurs d'entrée sont visualisées en bleu sur l'écran.

Exemple :

```
#ID
```

ENTREE (code courant d'entrée : décimal)

```
#8000                (8000 en decimal)
#8000H ou #$8000     (8000 en hexadecimal)
```

```
#IH
```

ENTREE (code courant d'entrée : hexadécimal)

```
#8000                (8000 en hexadecimal)
#&8000 ou #8000T     (8000 en decimal)
```

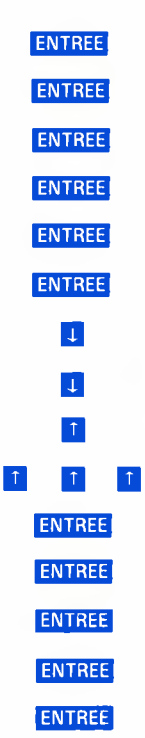
(ADRESSE)/ : Affichage et/ou modification du contenu d'une adresse mémoire ou des registres internes du microprocesseur 6809

Commande : Le moniteur en état « attente de commande » « # »
Tapez : <expression> / <contenu> <expression> ↑ ou ↓ ou ENTREE
ou : = <contenu> <expression> ↑ ou ↓ ou ENTREE

Effet : La commande (ADRESSE) / permet d'afficher et/ou de modifier le contenu de la case mémoire dont l'adresse est spécifiée dans la commande.
Si l'adresse est omise, le moniteur pointera sur la dernière adresse spécifiée.
Si aucune adresse n'a été spécifiée auparavant, le moniteur pointera sur l'adresse 0000.
L'adresse peut être définie par une étiquette contenue dans la table des symboles issue du dernier assemblage, ou par les symboles représentant les registres internes du 6809 (PC, A, B, D, DP, CC, X, Y, U, S).
Le nombre d'octets examinés à partir de l'adresse pointée dépend du mode de sortie.
En mode mnémonique, le nombre d'octets examinés peut être de un à cinq. En mode numérique le nombre d'octets dépend de la valeur du paramètre de la commande N (voir plus loin). Le caractère « = » seul visualise le contenu de la dernière adresse pointée.
Après le caractère « / » qui indique la fin de l'adresse, le moniteur affiche le contenu de la mémoire ou du registre.
Remarquez qu'un 0 précède tous les nombres hexadécimaux commençant par un caractère alphabétique, ceci pour éviter la confusion entre les nombres hexadécimaux et les registres du microprocesseur.
Après l'affichage du contenu de la mémoire, le système attend l'entrée d'une nouvelle valeur destinée à remplacer la précédente.
Cette entrée est facultative.
La touche ↓ permet d'examiner les octets suivants sans revenir au point « attente de commande ».
La touche ↑ permet d'examiner les octets précédents, mais par décrément de un octet uniquement.
La touche ENTREE termine la commande et retourne au point « attente de commande ».
Exemple :

```
#9C14/ LEAY  -1,Y
ENTREE
```

Remarque : Si les modes d'entrée et de sortie ne sont pas définis, le moniteur force l'**entrée en hexadécimal** et la **sortie en mode mnémonique.**




```
#N1     (nombre:1 octet)
#OH     (sortie numerique hexadecimale)
#9C14/  31      -
#A      (sortie en code ASCII)
#9C14 /  1     -  ( le code ASCII de 1 =31)
#N2     (nombre octets:2)
#9C14/   313F       -
 9C16/   26FA
 9C18/   GETPT
 9C17/   0FA10
 9C14/   313F      1836
#9C14/   1836      313F
#9C14/   313F
#= /31(le moniteur Pointe sur l'adresse 9C14
#PC/     RETMON   BC00 (PC est charge a BC00
#PC/     0BC00
```

MODE NUMERIQUE

Commande: Le moniteur en état « attente de commande » « # »

Tapez : N1 ENTREE
ou : N2 ENTREE
ou : N ENTREE

Effet : La commande N[n] (MODE NUMERIQUE) force la sortie du moniteur en mode numérique.
Le suffixe [n] {1 ou 2} représente le nombre d'octets à examiner.
Si [n] est omis dans la commande, le moniteur force n = 1.

Exemple :

```
                #IH
ENTREE
                #OH
ENTREE
                #N1
ENTREE
                #9C00/    0B6
ENTREE
                #N2
ENTREE
                #9C00/    0B6A7
ENTREE
                #N
ENTREE
                #9C00/    0B6
ENTREE
```

OUTPUT : sortie

Commande : Le moniteur en état « attente de commande » « # »
Tapez : 0D ENTREE
ou : 0H ENTREE

Effet : La commande O {H ou D} (OUTPUT) spécifie le mode numérique de sortie du moniteur
Les codes admis sont :

DÉCIMAL : D
HEXADÉCIMAL : H

Si la commande n'a pas été précisée, le moniteur forcera le code **hexadécimal** par défaut.
Sans suffixe, la valeur affichée est exprimée en hexadécimal.
Le code décimal est spécifié par le suffixe T.

Exemple :

```
        #N2
ENTREE
        9C00/    0B6A7
ENTREE
        #0D
ENTREE
        9C00/    44611T
ENTREE
        #0H
ENTREE
        9C00/    0B6A7
ENTREE
```

MODE ASCII

Commande : Le moniteur en état « attente de commande » « # »
Tapez : A ENTREE

Effet : La commande A (MODE ASCII) force la sortie du moniteur en mode caractère ASCII.
Le moniteur affichera le caractère représenté, en code ASCII, par le contenu de l'octet spécifié.
Si la valeur de l'octet n'est pas comprise entre 32 et 127 en décimal ou 2Ø et 7F en hexadécimal, représentant les caractères dits « imprimables », le moniteur affichera un point.

Exemple :

MODE MNEMONIQUE

Commande : Le moniteur en état « attente de commande » « # »
Tapez : M ENTREE

Effet : La commande **M (MODE MNEMONIQUE)** permet de reconstituer à la sortie du moniteur les instructions en langage symbolique du programme source. Cette opération est aussi appelée « désassemblage ».
Pour cela, le moniteur examine la mémoire par paquets de un à cinq octets suivant l'instruction.
L'adresse de la commande doit pointer sur le premier octet d'une instruction, sinon le désassemblage du programme objet n'aurait aucun sens.
Si le moniteur ne peut interpréter le contenu d'un octet, il affichera un point d'interrogation « ? »
Si aucun mode de sortie du moniteur n'est demandé, le système forcera le mode mnémonique.

Exemple :

```
                #M
ENTREE
                #9C00/      LDA     >PORTA
  ↓
                 9C03/      ORA     #1
  ↓
                 9C05/      STA     >PORTA
  ↓
                 9C08/      LDX     #STADR
  ↓
                 9C0E/      LDY     #20
```

IMPRESSION SYMBOLIQUE

Commande : Le moniteur en état "attente de commande" "#"
Tapez : **Z** ENTREE

Effet : La commande **Z** (symbolique) fixe, en symbolique, le mode courant d'impression des étiquettes et valeurs numériques.
Si le mode courant d'impression n'est pas spécifié, le mode symbolique est le mode pris par défaut.

Exemple :

IMPRESSION NUMÉRIQUE

Commande : Le moniteur en état "attente de commande" "#"
Tapez : **U** ENTREE

Effet : La commande **U** (numérique) fixe, en numérique, le mode courant d'impression des étiquettes et valeurs numériques.
L'impression numérique doit être spécifiée, car l'impression en symbolique est l'impression prise par défaut.

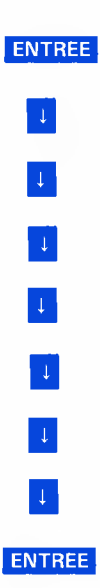
Exemple :

IMPRESSION MI-SYMBOLIQUE MI-NUMÉRIQUE

Commande : Le moniteur en état "attente de commande" "#"
Tapez : **H** ENTREE

Effet : La commande **H** (**HALF** mi-symbolique mi-numérique) fixe, en symbolique, l'impression des symboles situés en "zone adresse". Les symboles ou valeurs numériques situés en "zone opérande" sont affichés en numérique.

Exemple :




```
#H
# 9C00/     LDA     >0A7C0
  9C03/     ORA     #1
  9C05/     STA     >0A7C0
  9C08/     LDX     #0
  M2/       LDY     #14
  9C0F      LDD     #5555
  M0/       STD     ,X++
  9C14      LEAY    -1,Y
#
```

INDIRECTION

Commande : Tapez : → après une valeur d'adresse dont vous souhaitez connaître le contenu.

Effet : La commande → (INDIRECTION) permet d'introduire un niveau d'indirection dans l'examen du contenu des mémoires.
La commande INDIRECTION utilise le contenu de la sortie courante pour adresser la nouvelle mémoire.
Si la sortie est en mode numérique sur un octet ou en mode ASCII, le moniteur complétera l'adresse avec l'octet suivant.
En sortie sur deux octets, la valeur représente l'adresse à examiner.
En mode mnémonique, le moniteur pointera l'adresse indiquée dans la ligne courante. Si aucune adresse n'est présente ou si l'instruction ne touche pas au PC, la commande INDIRECTION est sans effet et répète la ligne courante.
Cette commande est très utile pour examiner la destination des branchements, par exemple : JSR, JMP, BRA, etc.

Remarque : si l'instruction touche au PC, la commande → visualise le contenu de l'adresse du PC après modification par cette instruction.

Exemples :

```
                #N2          (mode numérique 2 octets)
ENTREE
                #9C00/   0B6A7
   →
                B6A7/   5F53
   →
                5F53/    RETCAR
ENTREE
```

```
                #N1          (mode numérique 1 octet)
ENTREE
                #9C00/ 0B6
   →
                 0B6A7 / 5F
   →
                 5F53 / RETCAR
ENTREE
```

```
#M  (mode mnémonique)
```
ENTREE

```
#9C03/   ORA    #1
```
→

```
 9C03/   ORA    #1
```

(l'indirection est sans effet la ligne courante ne contient pas d'adresse)

↓

```
 9C05/   STA    >PORTA
```
→

```
PORTA/   SUBA   #90
```
→

```
PORTA/   SUBA   #90
```
ENTREE

```
#X/     RETCAR      9C00   (chargement de X)
```
ENTREE

(utilisation du contenu de X comme adresse d'indirection)

```
#X/ START
```
→

```
START/   LDA    >PORTA
```
ENTREE

MODE CALCULATEUR

Commande : Le moniteur en état « attente de commande » « # »
Tapez : <expression> =
A droite du signe « = », le résultat du calcul de l'expression sera affiché.

Effet : Le **MODE CALCULATEUR** du **MONITEUR** permet d'évaluer une expression et d'afficher le résultat.
Toutes les expressions admises par l'assembleur sont évaluables par le moniteur.
Les étiquettes présentes dans l'expression à évaluer doivent être également présentes dans la table des symboles issue du dernier assemblage.
L'évaluation d'une expression peut être utilisée pour examiner la mémoire correspondante.
Les résultats négatifs sont exprimés en valeur complémentaire plus 1. Cette manière de représenter les nombres est appelée « complément à 2 »
Par exemple :

1	ØØØ1
Ø	ØØØØ
-1	FFFF

```
#OH          (sortie en hexadécimal)
ENTREE
#IH          (entrée en hexadécimal)
ENTREE
#9C00+18=  9C18
ENTREE
#ENDMEM-START= 400
ENTREE
#OD          (sortie en décimal)
ENTREE
#ID          (entrée en décimal)
ENTREE
#1345+6640= 7985T
ENTREE
# 9C00H+05/ STA  >PORTA
```

GO : lancer

Commande : Le moniteur en état « attente de commande » « # »
Tapez : G <expression> ENTREE

Effet : La commande G <expression> (GO) permet de lancer l'exécution d'un programme à partir de l'adresse définie par l'expression de la commande.
La commande moniteur GO est équivalente à un saut à une adresse (instruction JMP).
Les étiquettes utilisées dans l'évaluation de l'expression doivent être présentes dans la table des symboles issue du dernier assemblage.
Si en cours d'exécution du programme le système rencontre l'instruction BD BØØØ, il affichera le message :
« 8 BRK @ (adresse) » et arrêtera l'exécution du programme.
Ce message (indice 8) indique qu'un **STOP** a été positionné par l'utilisateur dans son programme. Sous contrôle du moniteur, on ne peut placer que huit points d'arrêt numérotés de Ø à 7. **STOP** est utilisé pour retourner sous contrôle du moniteur à la fin de l'exécution du programme utilisateur.

Exemple :

#GSTART (exécution du programme « **MOIRAGE** »)

ENTREE

```
 8 BRK @ 9C2A
#
```

BREAK : arrêt

Commande : Le moniteur en état « attente de commande » « # »
Tapez : K <expression> ENTREE

Effet : La commande K <expression> (BREAK) positionne un point d'arrêt sur l'adresse spécifiée par l'expression de la commande. L'utilisateur a la possibilité de positionner huit points d'arrêt numérotés de 0 à 7 dans l'ordre de la programmation.
Le point numéro 8 est affiché par le moniteur quand il rencontre l'intruction BD BØØØ dans le programme utilisateur.
L'adresse de positionnement des points d'arrêt peut être définie par une constante, une étiquette ou une expression.
Les étiquettes utilisées doivent être présentes dans la table des symboles issue du dernier assemblage.
Un retour sous contrôle de l'éditeur ou MENU efface les points d'arrêt programmés précédemment.
Un point d'arrêt qui n'est pas positionné sur le premier octet d'une instruction va détruire un octet de cette instruction et provoquer des effets imprévisibles à l'exécution !!
IL NE FAUT JAMAIS FAIRE CELA !!!
La technique des points d'arrêt consiste à insérer une interruption logicielle SWI sur chaque adresse pointée par un point d'arrêt.
On ne peut donc programmer un point d'arrêt dans le SYSTEME, qui est en mémoire ROM.

Exemple :

```
          #K9C05
ENTREE
          #GSTART
ENTREE
            0 BRK @  9C05
          #
```

CONTINUE : continuer

Commande : Le moniteur en état « attente de commande » « # » après affichage d'un point d'arrêt.
Tapez : **C [n]** ENTREE

Effet : La commande **C [n] (CONTINUE)** permet de continuer l'exécution d'un programme suspendue par un point d'arrêt.
[n] indique le nombre de passages par le point d'arrêt qu'il faut faire avant que le programme y soit suspendu. A chaque point d'arrêt est associé un "nombre de passage" qui est initialisé à 1 par l'instruction **K**.
Si le paramètre n est omis, 1 est pris par défaut et le programme s'exécutera jusqu'au prochain point d'arrêt défini par l'utilisateur ou le moniteur.
Si le dernier point d'arrêt a été positionné sur un branchement vers une zone de programme en ROM, le MONITEUR affichera le message « Can't continue » (ne peut pas continuer) (voir commande BREAK)
Remarque : la commande **G (GO)** remet à 1 les nombres de passages associés aux **BREAKPOINTS**.

Exemple :

```
        #KM2
ENTREE
        #GSTART
ENTREE
         0 BRK @ M2
        #C
ENTREE
         0 BRK @ M2
        #
```

BREAKPOINT : point d'arrêt

Commande : Le moniteur en état « attente de commande » « # »
Tapez : B ENTREE

Effet : La commande B (BREAKPOINT) affiche les points d'arrêt programmés ainsi que leur adresse.
L'affichage est effectué dans l'ordre croissant des numéros.

Exemple :

```
#K9C03
ENTREE
#K9C08
ENTREE
#K9C0D
ENTREE
#K9C0F
ENTREE
#B
ENTREE
 0 BRK @  9C03
 1 BRK @  9C08
 2 BRK @  M2
 3 BRK @  9C0F
#
```

REGISTRE : registre

Commande : Le moniteur en état « attente de commande » « # »
Tapez : R ENTREE

Effet : La commande R(REGISTRE) permet d'afficher le contenu des registres internes du microprocesseur 68Ø9.
La ligne Ø de l'écran, sous contrôle du moniteur, spécifie le nom des registres.
Le moniteur force l'affichage des registres en hexadécimal quel que soit le mode de sortie.

Exemple :

```
 PC    A    B    DP    CC    X    Y    U    S

 #
```

```
#K9C05
ENTREE

#GSTART
ENTREE
 Ø BRK @ Ø9C05
#R

ENTREE  9C05 81 00 00 88 0000 0000 0000 2303
#
```

YANK : supprimer

Commande : Le moniteur en état « attente de commande » « # »
Tapez : Y[n] ENTREE
ou : Y ENTREE

Effet : La commande Y [n] (YANK) supprime le point d'arrêt numéroté n.
Si [n] est omis dans la commande, tous les points d'arrêt programmés précédemment seront supprimés.

Exemple :

```
#KM2
ENTREE
#KM1
ENTREE
#KM0
ENTREE
#K9C21
ENTREE
#B
ENTREE
 0 BRK @ M2
 1 BRK @ M1
 2 BRK @ M0
 3 BRK @  9C21

#Y2                    (supprime le point d'arrêt n° 2)
ENTREE
#B
ENTREE
 0 BRK @ M2
 1 BRK @ M1
 3 BRK @  9C21

#Y                     (supprime tous les points d'arrêt)
ENTREE
#B
ENTREE
#                      (plus de points d'arrêt)
```

DUMP : sortie de l'image mémoire

Commande : Le moniteur en état « attente de commande » « # »

Tapez :	D <expression1> <expression2>	ENTREE
ou :	D (expression)	ENTREE
ou :	D	ENTREE

Effet : La commande D<adresse1> <adresse2> (DUMP) permet de visualiser, en hexadécimal et ASCII, le contenu du bloc de mémoire compris entre <adresse1> et <adresse2>.
Si les expressions spécifiant <adresse1> et <adresse2> sont omises dans la commande, le moniteur visualisera les cent vingt-huit octets qui suivent l'adresse courante.
Si l'expression spécifiant <adresse2> est omise dans la commande, le moniteur visualisera les cent vingt-huit octets qui suivent <adresse1>.
<adresse1> et <adresse2> peuvent être définies par une constante, une étiquette ou une expression.
Le moniteur affichera seulement en ASCII les caractères dont le code ASCII est compris entre 20 et 7F et forcera un point [.] dans le cas contraire.

Exemples :

```
#D BB8A BBAA
```

ENTREE

```
BB8A  C6 02 17 FF 63 C6 04 D7  ....c...
BB92  B1 35 82 36 38 30 39 20  .5.6809
BB9A  4C 61 6E 67 75 61 67 65  Language
BBA2  20 4D 6F 64 75 6C 65 20   Module
BBAA  31 2E B0 50 61 67 65 A0  1..Page.
```

```
#D9C00
```

ENTREE

```
9C00  B6 A7 C0 8A 01 B7 A7 C0  ........
9C08  8E 00 00 10 8E 00 14 CC  ........
9C10  55 55 ED 81 31 3F 26 FA  UU..1?&.
9C18  10 8E 00 14 CC AA AA ED  ........
```

```
9C20  81 31 3F 26 FA 8C 1F 40  .1?&...@
9C28  23 E1 BD B0 00 00 00 00  #.......
9C30  00 00 00 00 00 00 00 00  ........
9C38  00 00 00 00 00 94 00 00  ........
9C40  00 00 00 00 00 00 00 00  ........
9C48  00 00 00 00 00 00 00 00  ........
9C50  00 00 00 00 00 00 00 00  ........
9C58  00 00 00 00 00 00 00 00  ........
9C60  00 00 00 00 00 00 00 00  ........
9C68  00 00 00 00 00 00 00 00  ........
9C70  00 00 00 00 00 00 00 00  ........
9C78  00 00 00 00 00 00 00 00  ........
```

```
#DM2 M0
```

ENTREE

```
9C0B  10 8E 00 14 CC 55 55 ED  .....UU.
```

```
#DSTART START+30
```

ENTREE

```
9C00  B6 A7 C0 8A 01 B7 A7 C0  ........
9C08  8E 00 00 10 8E 00 14 CC  ........
9C10  55 55 ED 81 31 3F 26 FA  UU..1?&.
9C18  10 8E 00 14 CC AA AA ED  ........
9C20  81 31 3F 26 FA 8C 1F 40  .1?&...@
9C28  23 E1 BD B0 00 00 00 00  #.......
9C30  00 00 00 00 00 00 00 00  ........
```

MOVE : déplacement d'un bloc d'octets

Commande : Le moniteur en état "attente de commande" "#"
Tapez : **J** <**adr. source**> <**adr. destination**> <**n. octets**> ENTREE

Effet : La commande **J (MOVE)** transfert un bloc d'octets de l'adresse "source" vers l'adresse "destination". <**n. octets**> représente la longueur du bloc à déplacer.
Si l'adresse "source" est plus grande que l'adresse "destination", le transfert se réalisera en incrémentant l'adresse "source" jusque "source + n".
Si l'adresse "destination" n'est plus grande que l'adresse "source", le transfert se réalisera en décrémentant l'adresse "source + n" jusque "source". Cette façon de réaliser un **MOVE** est nécessaire pour admettre la possibilité de chevauchement des zones source et destination.
Une tentative de transfert dans une zone de mémoire ROM est, bien entendu, sans effet et n'est pas signalée par un message du système.
Comme la commande **MOVE** ne teste pas l'adresse "destination", il faut prendre soin de ne pas écraser une zone RAM utile par ailleurs.
Exemple : le programme source, la table des symboles, la page ØØ, etc...
La commande **MOVE** est une commande à manier avec d'infinies précautions.

Exemple :

```
                #D5000 5007
ENTREE
                5000 00 00 00 00 00 00 00 00
                #D9C00 9C07
ENTREE
                9C00 B6 A7 C0 8A 01 B7 A7 C0
                #J 9C00 5000 8
ENTREE
                #D5000 5007
ENTREE
                5000 B6 A7 C0 8A 01 B7 A7 C0
                #
```

TRACE

Commande : Le moniteur en état « attente de commande » « # » après affichage d'un point d'arrêt.

Tapez : T[n] ENTREE

ou : T ENTREE

Effet : A partir d'un point d'arrêt, la commande T [n] (TRACE) permet de visualiser [n] instructions du programme en cours d'exécution ainsi que le contenu des registres internes du 6809.

Si [n] est omis dans la commande TRACE, le moniteur forcera n = 1.

Les instructions du programme ne sont visualisables qu'en mnémoniques.

La commande TRACE utilisant la technique des points d'arrêt, elle ne peut pas être utilisée sur la mémoire ROM (voir BREAK).

On ne peut pas « tracer » les routines du moniteur MO5. L'exécution en mode TRACE ne s'arrête pas sur un appel au moniteur. Seulement, évidemment, toute la partie contenue dans le moniteur n'est pas tracée. Le TRACE reprend à l'instruction suivante du programme utilisateur. (Instruction suivant le CALL Nombre pour un CALL et instruction dont l'adresse est au sommet de pile pour un GOTO.

Exemple :

```
   PC     A     B     DP     CC     X     Y     U     S

   #
   #KM2
ENTREE
   #GSTART
ENTREE
    0 BRK @ M2
   #T
ENTREE
    9C0F 81 00 00 80 0000 0014 0000 2303
         0BC14   LDD    #5555
   #T3
ENTREE
    9C12 55 55 00 80 0000 0014 0000 2303
         M0        STD    ,X++
    9C14 55 55 00 80 0002 0014 0000 2303
         9C14      LEAY   -1,Y
    9C16 55 55 00 80 0002 0013 0000 2303
         9C16      BNE    M0
```

WRITE : écrire

Commande : Le moniteur en état « attente de commande » « # » après affichage d'un point d'arrêt.
Tapez : W[n] ENTREE
ou : W ENTREE

Effet : La commande W[n] (WRITE) est une variante de la commande TRACE. La fonction est presque semblable, mais WRITE s'exécute sans afficher les sous-programmes qu'ils soient en mémoire RAM ou ROM.
WRITE est un TRACE du programme principal.
Il permet par ailleurs de tracer un programme dont les sous-programmes sont en ROM.
Les instructions des sous-programmes ne sont pas comptées dans la valeur [n].

Exemple :

```
   PC     A     B     DP     CC      X     Y     U     S

   #

  #K9C00
ENTREE
  #GSTART
ENTREE
    0 BRK @ START
  #W

   9C03 81 00 00 88 0000 0000 0000 2303
ENTREE     9C03    ORA    #1
  #W
ENTREE
   9C05 81 00 00 88 0000 0000 0000 2303
        9C05   STA    >PORTA
  #W2
ENTREE
   9C08 81 00 00 88 0000 0000 0000 2303
        9C08   LDX    #RETCAR
   9C0B 81 00 00 84 0000 0000 0000 2303
        M2     LDY    #GETPT
  #
```

PRINT : imprimer

Commande : Le moniteur en état « attente de commande » « # »
Tapez : P ENTREE

Effet : La commande P (PRINT) permet de commuter l'affichage du MONITEUR sur l'imprimante.
Après avoir reçu la commande PRINT, le moniteur demande la confirmation de la commande en envoyant le message « Printer Ready Y/N? »
Si la réponse est positive, la commande PRINT est exécutée. Si la réponse est négative, la commande PRINT est annulée et le moniteur attend une nouvelle commande.
La commutation de l'affichage sur l'imprimante préserve le contenu de l'écran.
Pendant la période d'affichage sur l'imprimante, les échos des commandes à partir du clavier ne sont pas générés. Les lignes sont affichées, en bloc, après validation par la touche ENTREE.
Une seconde commande PRINT génère un saut jusqu'au sommet de la prochaine page sur l'imprimante et redonne l'affichage sur l'écran en effaçant le contenu précédent.

Exemple :

#P

ENTREE

```
Printer Ready Y/N?
```

Y

(l'affichage est commuté sur l'imprimante)

P ENTREE

(l'imprimante effectue un saut de page, l'écran est effacé et l'affichage est redonné à l'écran)

SAVE : Sauvegarde d'un programme objet

Commande : Le moniteur en état « attente de commande » « # »
Tapez : S<descripteur> <adr.1> <adr.2> <adr.3> ENTREE

Effet : La commande S <descripteur de fichier> <adresse1> <adresse2> <adresse3> (SAVE) permet de sauvegarder un programme objet ou des données sur le périphérique assigné dans la commande.
Si le descripteur de fichier ne contient pas de suffixe, le moniteur forcera le suffixe .BIN par défaut.
La désignation du périphérique n'est pas obligatoire dans la commande. Le lecteur de disquettes Ø ou le LEP seront utilisés par défaut.
La zone mémoire à sauvegarder est comprise entre <adresse1> et <adresse2>, <adresse1> spécifiant le début et <adresse2> la fin de la zone mémoire à sauvegarder.
<adresse3> est l'adresse qui sera utilisée pour lancer l'exécution du programme.
<adresses 1, 2 et 3> peuvent être spécifiées par une constante, une étiquette ou une expression, sous réserve que les étiquettes utilisées soient présentes dans la table des symboles issue du dernier assemblage.
Si vous utilisez un LEP, le MONITEUR posera la question : « Cassette Ready Y/N? ». Si la réponse est positive le programme sera sauvegardé.

Exemple :

```
#S0:MOIRAGE1.OBJ 9C00 9C2A 9C00
```

ENTREE

(MOIRAGE1.OBJ est sauvegardé.)
La même commande peut aussi s'écrire :

```
#SMOIRAGE1 START START+2A START
```

ENTREE

(MOIRAGE 1.BIN est sauvegardé.)

LOAD : Chargement d'un programme objet

Commande : Le moniteur en état « attente de commande » « # »
Tapez : L<descripteur> [<offset>] ENTREE

Effet : La commande L <descripteur de fichier> <offset> (LOAD) charge en mémoire un programme objet qui a été sauvegardé sur le périphérique assigné dans la commande.
<offset> est optionnel dans la commande LOAD et permet une translation du programme objet.
Le moniteur charge le compteur de programmes (PC) avec l'adresse d'exécution qui a été spécifiée à la sauvegarde du programme.
Si (offset) est présent dans la commande, le compteur de programmes (PC) sera chargé à la valeur obtenue en ajoutant l'offset à l'adresse d'exécution.
Note : L'adresse d'exécution du programme peut être définie, dans le moniteur, par <adr3> dans la commande SAVE, ou bien dans l'éditeur, quand le programme objet est créé par l'assembleur ; l'adresse d'exécution est alors définie par la présence de l'étiquette, dans le champ opérande de la directive END, qui adresse la première instruction à exécuter dans le programme.
Si aucune étiquette ni symbole n'est spécifié avec la directive END, l'assembleur charge ØØØØ comme adresse d'exécution.

Exemple :

```
#L0:MOIRAGE1.OBJ
```

ENTREE

MOIRAGE1.ÒBJ est chargé à l'adresse spécifiée lors de la sauvegarde. (BCØØ)

```
#L0:MOIRAGE1 0040
```

ENTREE

MOIRAGE1.BIN est chargé à l'adresse spécifiée lors de la sauvegarde plus ØØ4Ø.(BC4Ø)

VERIFY : vérification

Commande : Le moniteur en état « attente de commande » « # »
Tapez : V<descripteur> <adr.1> <adr.2> <adr.3> ENTREE

Effet : La commande V <descripteur de fichier> <adresse1> <adresse2> <adresse3> (VERIFY) compare le contenu du fichier défini par le descripteur avec la zone mémoire comprise entre <adresse1> et <adresse2>.
<adresse1> début de zone
<adresse2> fin de zone
<adresse3> adresse d'exécution
Si les contenus sont les mêmes, le curseur se repositionne et attend une nouvelle commande moniteur.
Si la zone mémoire spécifiée et le fichier désigné sont différents, un message d'erreur « Verification error » est affiché sur la ligne commentaires du moniteur.
ATTENTION : adr1, adr2, adr3 doivent être EXACTEMENT les mêmes que ceux utilisés dans SAVE ou que le début, la fin et l'adresse d'exécution d'un fichier créé par l'assembleur. En outre, le programme assemblé vers un fichier peut ne pas correspondre à l'image mémoire s'il ne contient pas un ensemble de code CONTINU. Par exemple, des ORG successifs peuvent créer un fichier segmenté ; de même, des RMB incrémentent le compteur de programme sans créer de code. Dans ce cas, l'image mémoire n'est pas comparable au fichier du programme objet.

Exemple :

```
#L MOIRAGE1.OBJ
```
ENTREE
```
#V0:MOIRAGE1.OBJ 9C00 9C2A 9C00
```
ENTREE
```
#
```

QUIT : retour sous contrôle de MENU

Commande : Le moniteur en état « attente de commande » « # »
Tapez : Q ENTREE

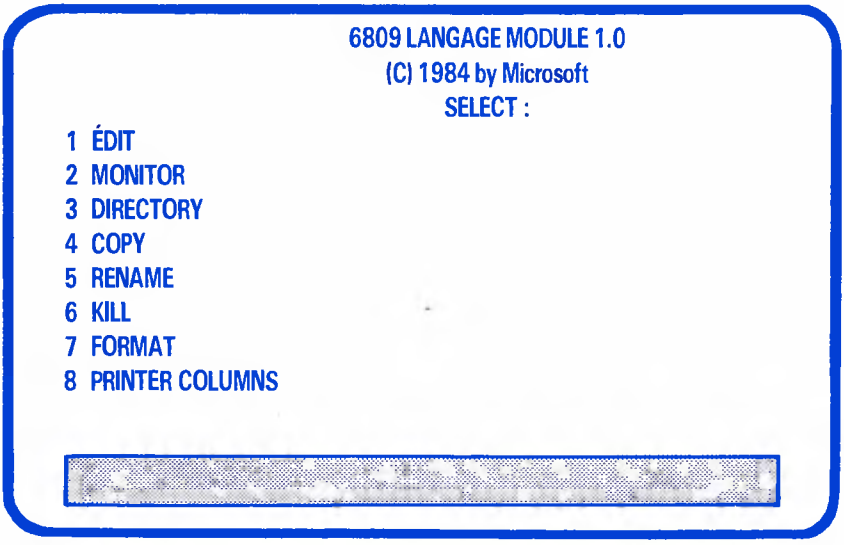
Effet : La commande Q (QUIT) redonne la main au MENU. À l'issue de la commande QUIT, le système génère l'écran de MENU et attend une commande.

Exemple :

#Q

ENTREE

MENU est affiché



EXIT : passage sous contrôle de l'ÉDITEUR

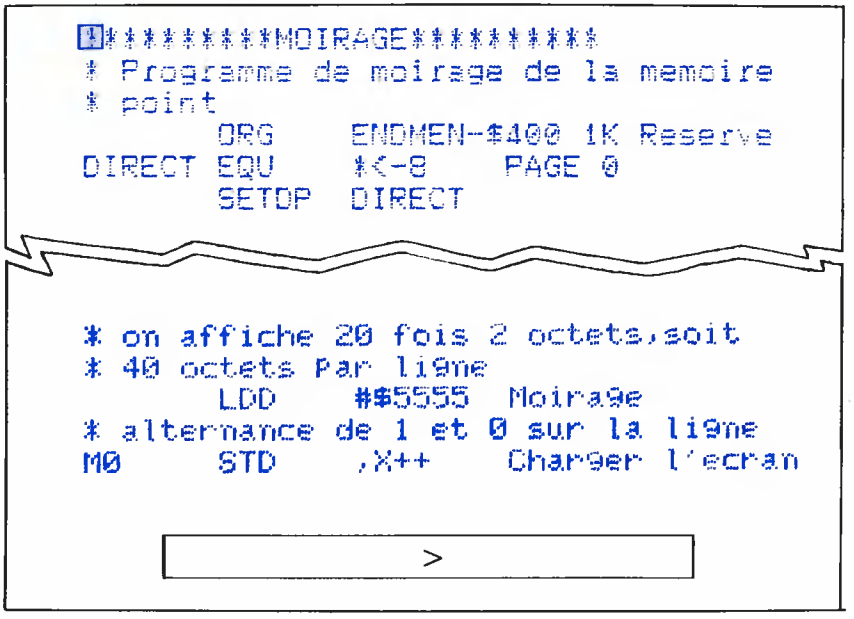
Commande : Le moniteur en état « attente de commande » « # »
Tapez : X ENTREE

Effet : La commande X (EXIT) permet au système de quitter le moniteur et de passer sous contrôle de l'éditeur.
A l'issue de la commande EXIT, le système génère l'écran de l'éditeur et affiche les vingt-trois premières lignes du programme source résidant.

Exemple :

```
# X
```

ENTREE



CHAPITRE 5
GESTION DE FICHIERS

La GESTION DE FICHIERS permet l'utilisation de fichiers sur disquettes et cassettes.
Il est constitué de six routines accessibles par le MENU qui permettent :
— d'établir le catalogue d'une disquette (DIRECTORY)
— de copier un fichier (COPY)
— de renommer un fichier (RENAME)
— d'effacer un fichier (KILL)
— de formatter une disquette (FORMAT)

DIRECTORY : catalogue
La commande s'applique seulement aux disquettes.
La commande de MENU 3 active la ligne DIRECTORY

3 DIRECTORY

Le système attend un numéro de lecteur de disquettes compris entre Ø et 3 puis une validation par la touche ENTREE .
Si aucun numéro de lecteur n'est précisé avant la commande ENTREE, le système utilise le lecteur Ø par défaut.
Un numéro non compris entre Ø et 3 provoque l'affichage du message « Bad parameter » (mauvais paramètre) et le retour sous contrôle de MENU.
Un numéro correspondant à un lecteur absent provoque l'affichage du message « Disk Not Ready » (disque non prêt) et le retour sous contrôle de MENU.
DIRECTORY (catalogue) affiche la liste des programmes enregistrés dans la disquette positionnée dans le lecteur spécifié.
La ligne 25 (rouge) est destinée à recevoir les commentaires et messages d'erreur sous contrôle de la routine DIRECTORY.

Exemple :
Disk not Ready ou Press any Key
(disque non prêt ou appuyer une touche pour continuer)

1. Nom du fichier (huit caractères maximum)

2. Suffixe
(par défaut).ASM Programme source
.BIN Programme objet
Note : Pour le nom du fichier et le suffixe, l'utilisateur peut choisir d'autres noms. Ces noms peuvent contenir tous les caractères imprimables (non accentués) sauf « < > » « ? » « » et « : »

3. Type de fichier.
.A Programme assembleur (source)
.M Programme machine (objet)

4. Type de donnée.
.A ASCII
.B Binaire

5. Nombre de blocs utilisés par le programme sur la disquette.
(taille d'un bloc = 1 K octet)

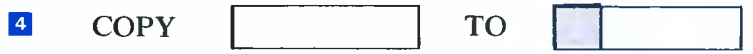
Le retour sous contrôle de MENU est obtenu en pressant une touche quelconque.

Note : La consultation de DIRECTORY n'efface pas les programmes source et objet en mémoire. Vous pouvez donc utiliser cette fonction, à tout moment, pour contrôler les enregistrements et modifications de nom de fichier que vous venez d'effectuer.
DIRECTORY vous permet aussi de savoir si une disquette est initialisée. Avec une disquette qui n'est pas formatée, le système retourne le message : « Unreadable Disk » (disque illisible).
La commande DIRECTORY permet également de connaître la place encore disponible sur la disquette en faisant :
Place disponible en K octets = 8Ø - $\lesssim$ blocs utilisés – 2

COPY : copie entre disquettes ou entre cassette et disquette

La commande de MENU 4 active la ligne COPY :

COPY attend le premier descripteur de fichier qui doit être obligatoirement le fichier à recopier.
La zone de saisie est limitée à quatorze caractères et validée par la commande ENTREE.
La ligne 25 (rouge) est destinée à recevoir les messages d'erreur.



La seconde zone, également limitée à quatorze caractères, est occupée par le fichier destination et validé par ENTREE.
Sa tâche effectuée, COPY retourne directement sous contrôle de MENU.
COPY permet la copie intégrale d'un fichier en changeant éventuellement son nom.
Avec deux lecteurs de disquettes, COPY permet de copier un fichier d'un lecteur sur l'autre en spécifiant les numéros des périphériques utilisés dans les descripteurs de fichier correspondants.
Avec un seul lecteur de disquette, COPY permet de copier un fichier sur la même disquette en changeant le nom du fichier. Si le périphérique n'est pas spécifié dans le descripteur source, le système utilisera le lecteur Ø par défaut. Les numéros de lecteurs autorisés sont : 1, 2 et 3.
Le suffixe est indispensable dans le descripteur de fichier source.
Le système complétera le descripteur destinataire avec les éléments du fichier source.
COPY permet également des copies de la cassette vers une disquette et vice-versa.

Exemple :

(COPIE DE Ø : MOIRAGE.RSM en 0 : MOIRAGE1.ASM)

(copie de Ø : MOIRAGE.ASM en C : MOIRAGE.ASM)

RENAME : changement de nom

La commande s'applique seulement aux disquettes.
La commande de MENU 5 active la ligne RENAME :

RENAME attend le premier descripteur de fichier qui doit être obligatoirement le fichier à renommer.
La zone de saisie est limitée à quatorze caractères et validée par la commande **ENTREE**.
La ligne 25 (rouge) est destinée à recevoir les messages d'erreur.

La seconde zone, également limitée à quatorze caractères, est occupée par le nouveau descripteur de fichier et validée par ENTREE .
Le changement de nom effectué, le système retourne sous contrôle de MENU.
La vérification du résultat peut être obtenu par la commande DIRECTORY.
RENAME permet le changement de nom d'un fichier sur une disquette seulement.
Le premier descripteur de fichier contient le nom du fichier à renommer, le second le nouveau nom du fichier.
Si le nouveau nom du fichier correspond à un fichier déjà existant dans la disquette, la message « File already exists » est affiché.
Si le périphérique n'est pas spécifié dans le descripteur à renommer, le système utilisera le lecteur Ø par défaut. Si le périphérique est omis dans le nouveau nom, le système utilisera le descripteur de fichier à renommer. Les numéros de lecteurs autorisés sont : Ø, 1, 2 et 3.
Le suffixe est indispensable dans le descripteur de fichier à renommer.

Exemple :

(renomme Ø : MOIRAGE.ASM en Ø : MOIRAGE1.ASM)

(renomme Ø : MOIRAGE1.ASM en Ø : MOIRAGE1.ESS)

KILL : suppression d'un fichier

La commande s'applique seulement aux disquettes.
La commande de MENU 6 active la ligne KILL :

KILL attend l'entrée du descripteur de fichier à détruire.
La zone de saisie est limitée à quatorze caractères et validée par la commande ENTREE
La ligne 25 (rouge) est destinée à recevoir les messages d'erreur ou les questions à destination de l'utilisateur. Après la validation de l'entrée, la commande KILL envoie, sur la ligne de commentaires, le message « Are you sure Y/N? » (êtes-vous sûr O/N ?).
Si la réponse est positive, le fichier est détruit, et le système retourne sous contrôle de MENU.
La commande KILL efface le fichier défini par le descripteur de fichier, supprime ses références dans le catalogue et libère la place qu'il occupe sur la disquette.
Si le périphérique n'est pas spécifié dans le descripteur de fichier à détruire, le système utilisera le lecteur Ø par défaut.
Le suffixe est obligatoire dans le descripteur de fichier à détruire.
La commande KILL ne fonctionne que sur un fichier résidant sur une disquette.

Exemple :

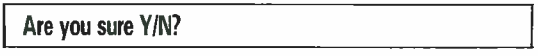
(créons le fichier « DETRUIT.FIC » afin de pouvoir le détruire dans la suite de notre exemple)

ENTREE

(Ø : DETRUIT.FIC est créé sur la disquette du lecteur Ø)

ENTREE

Le système répond



Y (Ø : DETRUIT.FIC est détruit)

FORMAT : initialisation d'une disquette

La commande s'applique seulement aux disquettes.
La commande de MENU 7 active la ligne FORMAT.

7 FORMAT

FORMAT attend l'entrée d'un numéro de lecteur de disquettes compris entre Ø et 3.
Cette entrée est obligatoire : contrairement aux autres commandes, le lecteur Ø n'est pas pris par défaut.
ENTREE valide le lecteur spécifié.
La ligne 25 (rouge) est destinée à recevoir les messages d'erreur ou les questions à destination de l'utilisateur. Après la validation de l'entrée, la commande FORMAT envoie sur la ligne de commentaires le message « Are you sure Y/N? » (« êtes-vous sûr O/N ? »)
Si la réponse est positive la disquette placée dans le lecteur spécifié est initialisée (on dit aussi formatée) et le système retourne sous contrôle de MENU.
La disquette ne doit pas être protégée, c'est-à-dire que son encoche de protection doit être libre.
Après la phase d'initialisation, le catalogue de la disquette est vide.
L'initialisation d'une disquette efface toutes les données qu'elle pouvait contenir auparavant.
Toute disquette neuve doit obligatoirement être initialisée avant d'être utilisée.

Exemple :
(nous vous proposons d'initialiser la disquette neuve que vous avez placée dans le lecteur Ø)

7 FORMAT Ø

ENTREE

Le système répond

```
Are you sure Y/N?
```

Y (Le lecteur Ø tourne. Dans trois minutes environ votre disquette sera initialisée ou formatée.)

PRINTER COLUMNS : déclaration du nombre de colonnes de l'imprimante

La commande de MENU 8 active la ligne PRINTER COLUMNS.

8 PRINTER COLUMNS 1 = 4Ø 2 = 8Ø

PRINTER COLUMNS attend un des deux chiffres représentant le nombre de colonnes de l'imprimante.

1 : 4Ø colonnes (type PR9Ø-Ø4Ø)
2 : 8Ø colonnes (type PR9Ø-Ø8Ø)

La ligne 25 (rouge) est destinée à recevoir les messages d'erreur.
Après l'entrée d'un des deux chiffres, le système retourne sous contrôle de MENU.
Un caractère différent de 1 ou 2 provoque l'erreur « Bad Parameter(s) » (mauvais paramètre).
Par défaut, le système force l'option 8Ø colonnes.
L'appel de la commande, avec ou sans modification du nombre de colonnes remet le compteur ligne à Ø, c'est-à-dire que la cartouche assembleur considère que l'imprimante est positionnée en haut d'une page.
Notez qu'il est possible de modifier, sous contrôle du moniteur, les registres commandant les paramètres de l'imprimante. Les valeurs par défaut sont :

— Le nombre de lignes de texte par page :
  64 lignes LINCNT 1 octet
— La taille totale d'une page :
  72 lignes PAGLEN 1 octet
  PAGLEN ≥ LINCNT
— La largeur d'une page :
  80 colonnes PAGWID 1 octet

Ces registres sont en RAM et peuvent être modifiés manuellement à partir du MONITEUR.
L'adresse en $BØØ3-BØØ4 pointe sur le registre LINCNT.
PEGWID est l'octet qui suit PAGLEN.

Format d'un fichier objet : sur une cassette ou une disquette.

1. Type de fichier dans le catalogue d'entrée est égal à 2.
2. Le fichier est composé d'une suite d'enregistrements de données suivie par un enregistrement d'adresse d'exécution.
3. Le format d'enregistrement de données est le suivant :

Décalage (octet)	Longueur (octet)	Description
0	1	type d'enregistrement = Ø
1	2	longueur d'enregistrement = N
3	2	adresse de chargement
5	N	données

4. Le format d'enregistrement d'adresse d'exécution est le suivant :

Décalage (octet)	Longeur (octet)	Description
0	1	type d'enregistrement = $FF
1	2	longueur d'enregistrement = 0000
3	2	adresse d'exécution

Avertissement

RESET doit être employé uniquement dans le cas de nécessité absolue. L'intégrité du système n'est pas garantie après l'utilisation de RESET. Il est conseillé de ne pas utiliser RESET avec une disquette en place dans un lecteur, car dans certains cas, le système peut tenter une écriture sur la disquette présente. De nombreuses heures de travail peuvent ainsi être réduites à néant !!

ANNEXE A

POINTS D'ENTREE DU MONITEUR MO5

1.0 INTRODUCTION

1.1 Le moniteur du MO5

Le moniteur du MO5 est formé de différentes "routines" (programmes) ayant chacune une fonction bien précise: gestion de l'écran, du clavier, lecture du crayon optique, des manettes de jeu, génération de musique, gestion de l'interface de communication, du lecteur-enregistreur de programmes, etc...

Les programmes d'application en ROM, comme le Basic résident ou l'Assembleur, utilisent ces routines pour leura propres besoins et vous pouvez faire de même. Le principe est le suivant:

1) Vous chargez avec des valeurs ayant une signification précise certains registres du 6809, et éventuellement certains registres en RAM (mots-mémoire sur un ou deux octets): ce sont les paramètres d'entrée.

2) Vous appelez la routine du moniteur qui effectue l'opération demandée, en utilisant le mécanisme explicité au § 1.2

3) Vous récupérez s'il y a lieu le résultat de l'opération dans certains registres du 6809, et/ou dans des registres en RAM: ce sont les paramètres de retour.

Dans certains cas, il n'y a pas de paramètres d'entrée, ou pas de paramètres de retour, mais le schéma général est toujours le même.

Dans ce qui suit, certains nombres ou adresses-mémoire seront écrits en hexadécimal pour plus de commodité: dans ce cas ils seront suivis par la lettre "H". Sauf avis contraire, tout nombre non suivi par un "H" est à lire en décimal.

Exemple: 10H = 16, 16H = 22, etc...

Les registres en RAM sur deux octets seront notés de la façon suivante: XXXXH-XXXXH où X représente un chiffre hexadécimal. Pour une meilleure compréhension, nous avons donné des noms à certains registres-mémoire: à chacun d'entre eux correspond l'adresse en mémoire du premier octet du registre.

Exemple: mettre la valeur 4B0H dans le registre 205AH-205BH signifie: mettre la valeur 04 dans le mot-mémoire d'adresse 205AH et mettre la valeur B0H dans le mot-mémoire d'adresse 205BH.

Les nombres en binaire seront précédés du symbole "%". Les bits notés bi sont numérotés de droite à gauche, le bit de poids le plus faible étant le bit 0, et celui de poids le plus fort étant le bit 7.

Exemple: %11001100

b7 = 1

b0 = 0

1.2 Accès aux routines du moniteur

L'accès normalisé à une routine du moniteur se fait en utilisant l'instruction "SWI" suivie d'un code (sur un octet). Ces codes sont au nombre de 58, répartis en deux groupes différenciés par l'état du bit 7, les 6 bits de poids faible formant le code proprement dit de la routine à laquelle on veut accéder. Le bit 7 a la signification suivante: s'il est à 0, un "JSR" à la routine eat simulé, c'est-à-dire que le retour se fait à l'instruction suivant l'appel; s'il est à 1, un "JMP" est simulé, c'est-à-dire que le retour se fera à l'instruction qui suit l'appel du programme où se trouve l'appel au moniteur.

EXEMPLE

```
PROG1   EQU     *
        SWI
        FCB     08H
        Instruction suivante  1
```

Le bip sonore se fera entendre, puis "instruction suivante 1" sera exécutée.

Mais si l'on rencontre dans un programme:

```
PROG1   EQU     *
        JSR     PROG2
        Instruction suivante  1
        . . . . . .

PROG2   EQU     *
        SWI
        FCB     88H
        Instruction suivante  2
```

Le bip sonore se fera entendre, puis "instruction suivante 1" sera exécutée, et non "instruction suivante 2".

L'instruction SWI sauvegarde toue les registree du 6809. Au retour, toue les registres sont remis dans l'état qu'ils avaient lors de l'appel, sauf le registre code condition du 6809 et bien sûr les registres devant contenir des paramètres de retour (le plue souvent B, X et Y).

Il vous est FORTEMENT CONSEILLE d'appeler les routines du moniteur en utilisent l'accès normalisé. En effet, ces routinea ne peuvent fonctionner correctement qu'avec le pointeur de pile et certains registres du 6809 dans un état bien précis.

La liste des codes des routines du moniteur est résumée dane le § 15.2.

2.0 GESTION DES INTERRUPTIONS

Les interruptions IRQ et FIRQ du 6809 aont programmables, c'est-à-dire qu'à chacune correspond un registre en RAM contenant l'adresse du programme qui doit les traiter. A la mise sous tension ou après un redémarrage "à chaud", ces registres ont été initialisés avec l'adresse d'un programme du moniteur.

Le moniteur utilise les interruptions IRQ (50 Hertz) pour gérer le clignotement du curseur et la répétition du clavier. Ces interruptions peuvent être aiguillées sur vos propres programmes en mettant dans le registre 2063H une valeur quelconque non nulle, et en mettant dans le registre TIMEPT (2061H-2062H) l'adresse de votre programme de traitement.

Pour gérer les FIRQ, vous devez mettre l'adresse de votre programme dans FIRQPT (2067H-2068H).

Attention cependant: certaines routines du moniteur sont interruptibles, et vos programmes ne doivent pas modifier leurs paramètres.

3.0 INITIALISATION

A la mise sous tension et après un "reset", le moniteur exécute un certain nombre de tests et d'initialisations:

3.1 Test de démsrrage à chaud ou à froid

Ce test est effectué pour ne pas modifier, en cas de redémarrage:

* le réglage du crayon optique,

* le mot de code de mise en mode graphique, spécifique de l'imprimante utilisée,

* la durée de la latence du clavier.

Et dans les deux cas:

Les registres d'aiguillage d'interruptions sont initialisés avec les adresses des routines moniteur correspondantes.

Les registres contenant les adresses des différentes tables (table de décodage du clavier, générateur de caractères standard, générateur de caractères utilisateur) sont réinitialisés de manière à pointer sur les tables standard.

Tous les autres registres sont remis aux valeurs standard (le plus souvent remis à zéro).

3.2 Initialisation du PIA système et du PIA jeux

Le PIA système est initialisé comme suit:

* port A:

  - entrée: bouton du crayon optique, lecture casaette.
  - sortie: sélection de la mémoire écran, écriture cassette, couleur du tour de l'écran.

* port B:

  - entrée: sortie clavier.
  - sortie: son, matriçage/dematriçage clavier.

* registre de contrôle:

  - entrée: cravon optioue, interruptions 50 Hertz.
  - sortie: moteur du LEP, commande d'inscrustation.

Les ports de données du PIA jeux sont initialisés en entrée.

4.0 GENERATEURS DE CARACTERES

4.1 Alphabet standard G0

Le générateur de caractères G0 est une suite de caractères affichables, correspondant au standard ASCII.

Chaque caractère est composé de 8 octeta, qui forment une matrice de 8x8 points représentant le caractère. Le premier octet du caractère correspond à la ligne inférieure de la matrice.

Exemple: matrice définissant le "A"

```
0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0
0 0 1 0 0 1 0 0
0 1 0 0 0 0 1 0
0 1 1 1 1 1 1 0
0 1 0 0 0 0 1 0
0 1 0 0 0 0 1 0
0 0 0 0 0 0 0 0
```

Dans le générateur, A sera donc défini par la liste d'octets suivants: 00H, 42H, 42H, 7EH, 42H, 24H, 18H, 00H

L'adresse du début du générateur G0 est contenue en RAM dans le registre GENPTR = 2073H-2074H. Vous pouvez redéfinir l'alphabet standard en créant votre propre générateur selon le principe énoncé ci-dessus, puis en chargeant le registre GENPTR avec l'adresse de ce générateur.

4.2 Alphabet G2

L'alphabet G2 contient 5 caractères, qui sont les accents aigu, grave et circonflexe, le tréma et la cédille. Ces caractères sont définis suivant le principe décrit ci-dessus, et destinés à être combinés avec des caractères minuscules.

4.3 Caractères utilisateur

Outre le registre contenant l'adresse du générateur standard, le registre USERAF = 2070H-2071H est destiné à contenir l'adresse de début d'un générateur de caractères utilisateur. Ce générateur sera organisé comme expliqué ci-dessus, 8 octets étant nécessaires pour définir un caractère. Quant aux codes, ils seront pris séquentiellement à partir de 80H. Cela veut dire que le code 80H correspondra au caractère défini par les 8 premiers octets, le code 81H au caractère défini par les 8 octets suivants, et ainsi de suite jusqu'au code FFH correspondant aux 8 derniers octets du générateur. Vous pouvez ainsi définir 128 caractères en plus de l'alphabet standard. L'adresse contenue par défaut dans le registre USERAF est celle du générateur G0.

5.0 PRIMITIVES DE GESTION D'ECRAN

Le MO5 peut gérer un écran pleine page de 320x200 points, avec 16 couleurs de forme, 16 couleurs de fond, et 16 couleurs de tour. En mode alphanumérique, il peut afficher 25 lignes de 40 caractères, définis par une matrice de 8x8 points, et ce dans une couleur au choix parmi 16, quelle que soit la couleur du fond. En mode graphique, l'unité centrale contrôle 8000 segments de 8 points chacun, et à chaque segment correspondent deux octets, situés à la même adresse logique. L'un est situé en mémoire dite "caractère", l'autre en mémoire dite "couleur", un bit de PIA permettant d'adresser physiquement l'une ou l'autre mémoire.

Dans un segment, un point peut prendre au choix deux couleurs: une couleur dite de "forme", auquel cas le bit correspondant dans l'octet en mémoire caractère est à 1, et une couleur dite de "fond", auquel cas le bit correspondant en mémoire caractère est à 0. L'octet se trouvant à la même adresse mais en mémoire "couleur" complète cette information: les 4 bits de poids fort donnent la couleur de la "forme", c'est-à-dire la couleur que devront prendre les points mis à 1, et les 4 bits de poids faible donnent la couleur du "fond", c'est-à-dire la couleur que devront prendre les points mis à 0. Les 4 bits donnant la couleur sont pBVR où p est le bit de pastel (couleurs claires, sauf pour le orange) et où B, V, R indiquent la présence de bleu, de vert, ou de rouge (les trois couleurs de base pour la T.V.) dans la couleur.

Il est donc parfois imprécis de parler de points "allumés" ou "éteints": l'on peut avoir des points clairs appartenant au "fond" (donc de codage 0) et des points foncés appartenant à la "forme" (donc de codage 1).

Pour mémoire, sachez que:

Rouge + vert = jaune

Rouge + bleu = magenta

Bleu + Vert = cyan

Rouge + vert + bleu = blanc

et bien sûr ni rouge, ni bleu, ni vert donne du noir.

Exemple: 1000

R = 0: pas de rouge

V = 0: pas de vert

B = 0: pas de bleu

p = 1: couleur claire

Le résultat est "noir clair", soit gris...

Autre exemple: 1111

R = 1: rouge présent

V = 1: vert présent

B = 1: bleu présent

p = 1: couleur claire

Le résultat devrait être "blanc clair". Cette couleur faisant double emploi avec le blanc, une exception a été faite: c'est le orange.

Ls MO5 offre également la possibilité d'incrustation de l'image T.V. dans l'image MO5: si votre MO5 est équipé de l'extension incrustation, la mise en mode "incrustation" (expliquée au § 5.4.4) laisaera apparaître l'image T.V. à la place de toutes les zones noires, comme si la couleur noire devenait transparente.

5.1 Mise en mémoire couleur

* code du point d'entrée: 04 pour un JSR, 84H pour un JMP.

Cette routine vous permet d'aller aélectionner la mémoire

couleur: tout ce que vous écrirez entre l'adresse 0 et l'adresse 1FFFH modifiera la couleur des points correspondants.

5.2 Mise en mémoire caractère

* code du point d'entrée: 06 pour un JSR, 86H pour un JMP.

Cette routine vous permet d'aller sélectionner la mémoire caractère: tout ce que vous écrirez entre l'adresse 0 et l'adresse 1FFFH modifiera l'état (fond ou forme) des points correspondants.

5.3 Bip sonore

* code du point d'entrée: 08 pour un JSR, 88H pour un JMP.

* paramètre d'entrée:

- registre STATUS (2019H)

Cette routine teste l'état du bit 3 du registre STATUS. S'il est à 0, elle génère le bip sonore, sinon elle retourne sans rien faire.

5.4 Gestion de caractères alphanumériques

* code du point d'entrée: 02 pour un JSR, 82H pour un JMP

* paramètre d'entrée:

- registre 6809 B

5.4.1 Séquence normale

5.4.1.1 Codes affichablea

Les codes affichables sont compris entre 20H et 7FH. Le caractère est affiché à la position courante du curseur. Si vous avez modifié le contenu du pointeur sur le générateur de caractères standard, les caractères affichés seront ceux que vous avez définis (entre 20H et 7FH).

5.4.1.2 Codes interprétables

Les codes interprétables sont compris entre 7 et 1FH. A la réception du code, l'une des opérations suivantes est effectuée:

- 07 : BEL--> "Bip" sonore.
- 08 : BS --> Déplacement d'une position vers la gauche ou recopie à gauche du caractère courant, si le registre COPCHR (202FH) contient la valeur FFH.
- 09 : HT --> Déplacement d'une position vers la droite ou recopie à droite du caractère courant, si le registre COPCHR (202FH) contient la valeur FFH.
- 0AH : LF --> Descente d'une ligne.
- 0BH : VT --> Remontée d'une ligne.
- 0CH : FF --> Effacement de toute la fenêtre.
- 0DH : CR --> Retour au début de la ligne courante.
- 0EH : SO --> Rien.
- 0FH : SI --> Rien.
- 10H : DLE--> Rien.

- 11H : DC1--> Allumage du curseur.
- 12H : DC2--> Rien.
- 13H : DC3--> Rien.
- 14H : DC4--> Extinction du curseur.
- 15H : NAK--> Rien.
- 16H : ACC--> Séquence accent.
- 17H : ETB--> Rien.
- 18H : CAN--> Effacement de la fin de la ligne à partir de la position courante du curseur.
- 19H : EM --> Rien.
- 1AH : SUB--> Rien.
- 1BH : ESC--> Séquence "d'échappement".
- 1CH : FS --> Rien.
- 1DH : GS --> Rien.
- 1EH : RS --> Retour du curseur dans le coin supérieur gauche de la fenêtre courante.
- 1FH : US --> Séquence de positionnement curseur ou de définition de fenêtre.

5.4.1.3 Caractères utilisateur

Les codes compris entre 80H et FFH sont ceux des caractères utilisateur. Avant d'appeler la routine pour faire afficher le caractère correspondant au code, vous devrez avoir chargé le registre USERAF avec l'adresse de votre générateur de caractères, comme expliqué au § 4.3.

5.4.2 Séquence de curseur ou de fenêtre

Le code 1FH définit une séquence de positionnement du curseur ou de définition de la fenêtre. Trois appels à la routine sont nécessaires, et peuvent être schématisés comme suit: US/N1/N2.

5.4.2.1 Positionnement du bas de la fenêtre (N1,N2 C <10H,19H>)

Cette séquence définit le bas de la fenêtre courante, où le numéro de ligne est égal à 10 x n1 + n2, n1 et n2 (chiffre des dizaines et chiffre des unités) étant respectivement les quatre bits de poids faible de N1 et N2.

EXEMPLE:

Pour définir le bas de la fenêtre en ligne 21, l'on doit envoyer dans B:

1° appel: code 1FH dans B

2° appel: code 12H dans B

3° appel: code 11H dans B

5.4.2.2 Positionnement du haut de la fenêtre (N1,N2 C <20H,29H>)

Cette séquence définit le haut de la fenêtre courante, où le numéro de ligne eat égal à 10 x n1 + n2, n1 et n2 étant respectivement les quatre bita de poids faible de N1 et N2.

5.4.2.3 Positionnement du curseur (N1,N2 C <40H,70H>)

Le curseur est positionné en ligne n1, colonne n2, où n1 et n2 sont respectivement les 6 bits de poids faible de N1 et N2, avec n1 compris entre 0 et 24 et n2 compris entre 1 et 40.

EXEMPLE:

Pour positionner le curseur en ligne 3, colonne 20, il faudra mettre successivement dans B avant chaque appel:

- 1° appel: code 1FH dans B
- 2° appel: code 43H dans B
- 3° appel: code 54H dans B

5.4.3 Séquence accent

La séquence ACC est utilisée pour permettre l'affichage d'une minuscule accentuée. On peut schématiser cette séquence comme suit: ACC/CODE/L.

1° appel: envoyer ACC, code d'une séquence accent (16H)

2° appel: envoyer CODE qui est le code de l'accent :

41H pour l'accent aigu

42H pour l'accent grave

43H pour l'accent circonflexe

48H pour le tréma

4BH pour la cédille

3° appel: envoyer L qui est le code de la minuscule à accentuer. Si ce code ne correspond pas à un code de minuscule, alors la lettre dont le code est L est affichée, et l'accent est effacé.

5.4.4 Séquence "d'échappement"

Cette séquence est utilisée pour positionner les attributs de couleur de l'écran, et d'autres attributs de la vidéo comme la taille des caractèrea, la mise en mode incrustation pour les MO5 équipés de l'extension correspondante, le type de scroll, etc... Ces attributs peuvent être de deux types: courant ou plein écran. Dans le cas d'attributs de type courant, la séquence est de la forme: ESC/ATT, et dans le cas d'attributs de type plein écran, la séquence est de la forme: ESC/ESPACE/ATT où ESC est le code de la séquence d'échappement, ESPACE le code de la barre d'espacement (20H), et ATT le code de l'attribut. Les codes d'attributs sont identiques, qu'ils soient de type courant ou plein écran, mais certains attributs (couleur du tour, attributs divers sauf la vidéo inverse) n'ont pas de signification en plein écran, et sont alors ignorés.

5.4.4.1 Attributs de couleur

Si ATT compris entre 40H et 6FH, la zone de l'écran qui sera modifiée est sélectionnée par les 4 bits de poida fort de l'octet "ATT", tandis que les 4 bits de poids faible sélectionnent l'une des 16 couleurs MO5.

* 4 bits de poids fort:

  4 --> modification de la couleur de la forme.

  5 --> modification de la couleur du fond.

  6 --> modification de la couleur du tour.

* 4 bits de poids faible:

00 --> noir

01 --> rouge

02 --> vert

03 --> jaune

04 --> bleu

05 --> magenta

06 --> cyan

07 --> blanc

08 --> gris

09 --> rouge clair

10 --> vert clair

11 --> sable

12 --> bleu clair

13 --> rose

14 --> bleu ciel

15 --> orange

EXEMPLE:

Après la séquence suivante, vous obtiendrez un fond orange:

```
LDB     #18H    code ESC
SWI
FCB     02
LDB     #5FH    5 pour le fond, F pour l'orange.
SWI
FCB     02
```

5.4.4.2 Attributs divers

Si ATT est compris entre 70H et 7FH, il a la signification suivante:

70H --> caractères en taille normale.

71H --> double hauteur, largeur normale.

72H --> double largeur, hauteur normale.

73H --> double taille.

74H --> écriture du caractère dans la couleur courante.

75H --> écriture du caractère sans modifier la couleur.

76H --> suppression de l'incruatation.

77H --> mise en mode incrustation.

78H --> scroll à vitesse normale.

79H --> scroll doux.

7AH --> mode page (pas de scroll).

7BH --> inversion de la vidéo.

Si un code supérieur à 7BH est envoyé, tout se passera comme si l'on avait envoyé le code 7BH. Si l'on envoie un code inférieur à 40H, rien ne se passera, de même que si l'on envoie dans une séquence plein écran le code d'un attribut qui n'a de sens qu'en tant qu'attribut courant.

EXEMPLE:

Après la séquence suivante, et si le MO5 est équipé de l'extension incrustation, toutes les zones noires de l'écran seront remplacées par l'image T.V.

```
LDB     #18H    code de la séquence d'échappement.
SWI
FCB     02
LDB     #77H    code de la mise en mode incrustation.
SWI
FCB     02
```

6.0 CLAVIER

6.1 Lecture rapide du clavier

* code du point d'entrée: 0CH pour un JSR, 8CH pour un JMP.

* paramètres de retour:

- registres 6809 A, B et CC

Cette routine effectue une lecture rapide du clavier, pour tester si une touche a été enfoncée ou non. Si aucune touche n'a été enfoncée, le bit Z du registre CC est mis à 1, sinon il est mis à 0. Dans ce cas, B contient le code de la touche enfoncée, et l'état des bits 0, 1 et 2 de A indique si la touche jaune, ou BASIC, ou CNT a été enfoncée:

- bit0: touche BASIC

- bit1: touche CNT

- bit2: touche JAUNE

Si le bit correspondant à la touche est à 1, la touche a été enfoncée, sinon le bit est à 0.

6.2 Décodage du clavier

* code du point d'entrée: 0AH pour un JSR, 8AH pour un JMP.

* paramètres d'entrée:

- regiatres CHRPTR (206DH-206EH), STATUS (2019H), LATCLV (2076H)

* paramètres de retour:

- registres 6809 B et CC

- registre KEY (2037H)

Cette routine effectue le décodage du clavier, et gère la répétition d'une touche après un certain délai. Ce délai est programmable via le registre LATCLV: en modifiant le contenu de ce registre, vous obtenez une latence du clavier. De la même facon, la répétition clavier peut être inhibée en chargeant ce registre avec une valeur suffisamment élevée. Le bip sonore qui se fait entendre chaque fois qu'une touche est frappée peut être inhibé lui-aussi, de deux façons: la première, classique, consiste à baisser le son de votre récepteur T.V., la seconde à forcer à 1 le bit 3 du registre STATUS.

B retourne soit le code ASCII du caractère, soit le code numérique de la touche (au caa où la touche BASIC a été enfoncée). Si aucune touche n'a été enfoncée, ou si l'une des trois touches BASIC, JAUNE, ou CNT a été enfoncée seule, ou si plusieurs touches parmi les touches BASIC, JAUNE, ou CNT ont été enfoncées simultanément, B retourne la valeur zéro.

La touche jaune sélectionne le caractère imprimé en jaune sur la touche. La touche CNT force à 0 le bit 6 du code ASCII du caractère imprimé en blanc sur la touche. La touche BASIC retourne le code numérique de la touche (qui dépend du matriçage du clavier), tout en forçant le bit 7 à 1 (codes retournés de 80H à B9H).

L'accès aux minuscules accentuées nécessite en général troia frappes consécutives:

- 1° frappe: touche accent.

- 2° frappe: touche jaune en même temps que l'accent.

- 3° frappe: lettre minuscule.

Cependant, l'accès à certainea minuacules accentuées couramment utilisées en Français peut être fait en deux touches:

* ACC/6: é

* ACC/7: è

* ACC/B: ù

* ACC/9: ç

* ACC/0: à

Mais dans tous les cas, vous devrez faire trois appels consécutifs à cette routine:

- le 1° appel retourne dans B le code de la touche ACC, soit 16H.

- le 2° appel retourne dans B le code de l'accent ou de la cédille.

- le 3° appel retourne dans B le code de la minuscule.

Remarque: bien que ACC suivi de 0 produise un à, et ACC suivi de touche jaune / 0 produise un accent grave, la frappe de touche jaune / 0 non précédée de la frappe de ACC retournera 0 dans B (pas de caractère connu), car la quote inverse n'est pas reconnue par le MO5.

Le code ASCII du caractère est lu dans une table dont l'adresse se trouve dans le registre CHRPTR. Si vous voulez redéfinir votre clavier, il suffira de mettre dans ce registre l'adresse de la table contenant vos propres codes ASCII, mais les séquences "accent" peuvent donner des résultats imprévisibles ...

Voici la liste des valeurs retournées quand la touche BASIC est enfoncée simultanément avec une touche autre que la touche jaune ou CNT:

COOE RETOURNE	TOUCHE ENFONCEE
82H	STOP
83H	ACC
85H	ENTREE
86H	RAZ
87H	C
88H	↑
89H	W
8AH	1
8BH	+
8CH	A
8DH	*
8EH	Q
8FH	V
90H	←
91H	X
92H	2
93H	-
94H	Z
95H	/
96H	S
97H	B
98H	↓
99H	Barre d'eepacement
9AH	3
9BH	0
9CH	E
9DH	P
9EH	D
9FH	M
A0H	→
A1H	@
A2H	4
A3H	9
A4H	R
A5H	O
A6H	F
A7H	L
A8H	⌐
A9H	.
AAH	5
ABH	8
ACH	T
ADH	I
AEH	G
AFH	K
B0H	INS
B1H	,
B2H	6
B3H	7
B4H	Y
B5H	U
B6H	H
B7H	J
B8H	EFF
B9H	N

7.0 PRIMITIVES GRAPHIQUES

7.1 Allumage ou extinction d'un point

* code du point d'entrée: 10H pour un JSR, 90H pour un JMP.

* paramètres d'entrée:

- registres 6809 X et Y

- registres FORME (2029H), CHDRAW (2036H), COLOUR (2028H)

* paramètres de retour:

- registres PLOTX (2032H-2033H), PLOTY (2034H-2035H)

7.1.1 Mode graphique

Cette routine met en couleur le point de coordonnées passées par X et Y, où X est compris entre 0 et 319 et Y entre 0 et 199. La couleur du point est déterminée par le contenu du registre FORME, qui doit être compris entre -16 et +15. Si le code de la couleur est négatif, le point sera écrit en "fond", c'est-à-dire que le bit correspondant dans l'octet en mémoire "caractère" sera mis à zéro. Si le code est positif, le point sera écrit en "forme", c'est-à-dire que le bit correspondant dans l'octet en mémoire "caractère" sera mis à 1. Si le bit 4 du registre STATUS est à 1, seul le bit en mémoire caractère sera modifié, la couleur ne sera pas changée.

Voici les codes des couleurs "forme" ou "fond":

COULEUR	CODE "FORME"	CODE "FOND"
NOIR	0	-1
ROUGE	1	-2
VERT	2	-3
JAUNE	3	-4
BLEU	4	-5
MAGENTA	5	-6
CYAN	6	-7
BLANC	7	-8
GRIS	8	-9
ROSE	9	-10
VERT CLAIR	10	-11
SABLE	11	-12
BLEU CLAIR	12	-13
PARME	13	-14
BLEU CIEL	14	-15
ORANGE	15	-16

Un code de couleur "fond" ae déduit donc d'un code de couleur "forme" en ajoutant 1 et en prenant l'opposé.

Le registre CHDRAW doit être mis à zéro, sinon l'on est dans le mode caractère, explicité ci-dessous.

7.1.2 Mode caractère

Si le contenu du registre CHDRAW est non nul, il est supposé être le code ASCII d'un "point caractère" à afficher à l'endroit défini par les registres X et Y. Dans ce cas, on doit avoir X C <1,40> et Y C <0,24>.

La couleur du "point-caractère" est fournie non plus par le registre FORME, mais par le registre COLOUR qui contient les couleurs courantes de "fond" et de "forme".

On peut accéder directement à l'écriture du caractère du registre CHDRAW en appelant la routine CHPL$ de numéro 12H pour un JSR ou 92H pour un JMP.

Dans les deux modes, les registres X et Y seront recopiés respectivement dans les registres PLOTX et PLOTY qui gardent l'abscisse et l'ordonnée du dernier point "allumé".

7.2 Tracé d'un segment de droite

* Code du point d'entrée: 0EH pour un JSR, 8EH pour un JMP.

* paramètres d'entrée:

- registres 6809 X et Y

- registres PLOTX (2032H-2033H), PLOTY (2034H-2035H), FORME (2029H), COLOUR (202BH), CHDRAW (2036H)

* peramètres de retour:

- registres PLOTX (2032H-2033H), PLOTY (2034H-2035H)

7.2.1 Mode graphique

Cette routine trace un segment de droite entre le point défini par son abscisse en PLOTX et son ordonnée en PLOTY (dernier point allumé si vous n'avez pas modifié ces registres depuis), et le point dont les coordonnées sont passées par X et Y, avec X compris entre 0 et 319 et Y entre 0 et 199. La couleur du segment est définie per le contenu du registre FORME, suivant les conventions explicitées ci-dessus.

Si le bit 4 du registre STATUS est mis à 1, la couleur ne sera pas mise à jour. Le contenu du registre CHDRAW doit être nul, sinon le segment tracé est un "segment caractère".

7.2.2 Mode caractère

Si le contenu du registre CHDRAW n'est pas nul, il est interprété comme le code ASCII du caractère servant au tracé. Dans ce cas, la couleur est ls couleur courante donnée par le registre COLOUR, et les coordonnées X et Y doivent être comprises respectivement entre 1 et 40 et 0 et 24.

Dans les deux modes, les registres X et Y seront recopiés respectivement dans les registres PLOTX et PLOTY qui gardent l'abscisse et l'ordonnée du dernier point allumé, ce qui permet de tracer un contour polygonal sans avoir à préciser à chaque fois le point de départ.

7.3 Segments horizontaux

Dans le cas où le segment à trecer est horizontel, un algorithme de tracé rapide est mis en oeuvre, ce qui peut être utile pour remplir des contours par une série de segments horizontaux.

7.4 Lecture de la couleur d'un point

* code du point d'entrée: 14H pour un JSR, 94H pour un JMP.

* paramètres d'entrée:

- registres 6809 X et Y

* paramètre de retour:

- regiatre 6809 B

Cette routine retourne dans l'accumulateur B la couleur d'un point : de -16 à -1 si le point est en "fond", et de 0 à +15 si le point est en "forme". Les coordonnées du point sont passées par X compris entre 0 et 319, et Y compris entre 0 et 199. La couleur est codée comme décrit au § 7.1.1

8.0 LECTURE DE L'ECRAN

* code du point d'entrée: 1AH pour un JSR, 9AH pour un JMP

* paramètres d'entrée:

- registres 6809 A et X

- registre SS3GET (205DH)

* paramètres de retour:

- registre 6809 B

- registres SS2GET (205CH), SS3GET (205DH)

Cette routine retourne dans l'accumulateur B le code ASCII du caractère dont les coordonnées eont passées par X compris entre 1 et 40, et A compris entre 0 et 24.

S'il n'y a pas de caractère connu en (X,A) alors B retourne la valeur zéro. Sinon, deux cas peuvent se produire:

8.1 Caractère normal

Si la caractère appartient à l'alphabet G0, alors B retourne le code ASCII du caractère reconnu, et les registres SS2GET et SS3GET sont remis à zéro.

8.2 Minuscule accentuée ou ç

Dans ce cas, trois appels à la routine sont nécessaires:

- 1° appel: une minuscule accentuée est détectée: B retourne le code ACC (accent), soit 16H, le registre SS3GET est chargé avec le code de l'accent (qui appartient à l'alphabet G2), et le registre SS2GET est chargé avec le code ASCII de la minuscule.

- 2° appel: B retourne le code de l'accent (qui se trouve maintenant dans le registre SS3GET), le registre SS3GET est chargé avec la valeur 80H, et le registre SS2GET n'est pas modifié.
- 3° appel: B retourne le code ASCII de la minuscule (qui se trouve dans le registre SS2GET), et les registres SS2GET et SS3GET sont remis à zéro.

ATTENTION: cette routine ne fonctionne correctement qu'avec l'alphabet standard GO. Si vous avez changé le contenu du registre GENPTR, vous devez restituer son ancien contenu avant d'appeler la routine, c'est-à-dire le charger avec l'adresse de début du générateur GO.

9.0 GENERATION DE MUSIQUE

* code du point d'entrée: 1EH pour un JSR, 9EH pour un JMP.

* paramètres d'entrée:

- registre 6809 B

- registres DUREE (203BH-203CH), OCTAVE (203EH-203FH), TIMBRE (203DH), TEMPO (2039H-203AH).

La note à jouer est passée par l'accumulateur B. Il y a 13 notes de base, de DO à UT, plus le silence. Voici la table des valeurs correspondant aux notes:

NOTE:	CODE:
SILENCE	00
DO	01
DO♯	02
RE	03
RE♯	04
MI	05
FA	06
FA♯	07
SOL	08
SOL♯	09
LA	0AH
LA♯	0BH
SI	0CH
UT	0DH

Il faut ensuite préciser les paramètres suivants: tempo, octave, timbre et durée.

* l'octave: il y a 5 octaves possibles, de l'octave 1 qui est la plus grave, à l'octave 5. L'octave 4 correspond à l'octave du LA 440.

Voici la liste des valeurs à mettre dans le registre OCTAVE:

OCTAVE	VALEUR
1	16
2	08
3	04
4	02
5	01

* la durée: il s'agit de la durée relative de chaque note, pouvant aller de la ronde à la triple croche. La valeur de la durée est à charger dans le registre DUREE. La valeur pour la ronde est 96, et l'on obtient les valeurs relatives en divisant par des puissances de 2, ou de 3 pour des triolets ou les notes pointées. Voici la liste des valeurs:

NOTES:	VALEURS:
RONDE	96
BLANCHE pointée	72
BLANCHE	48
NOIRE pointée	36
NOIRE	24
CROCHE pointée	18
CROCHE	12
DOUBLE CROCHE pointée	09
DOUBLE CROCHE	06
TRIPLE CROCHE pointée	05
TRIPLE CROCHE	03

DANS UN TRIOLET:	VALEURS:	
NOIRE	16	(car 3x16 = 48)
CROCHE	08	(car 3x8 = 24)
DOUBLE CROCHE	04	(car 3x4 = 12)
TRIPLE CROCHE	02	(car 3x2 = 06)

* le tempo: c'est le mouvement auquel doit être joué le morceau. La durée réelle d'une note est égale su tempo x la durée. La valeur du tempo, de 1 à 255, doit être chargée dana le registre TEMPO.

* le timbre: cette valeur (de 0 à 5) doit être chargée dans le registre TIMBRE. Le rapport cyclique est modifié en conséquence, ce qui donne à la note une attaque différente. Pour une note continue, il faut mettre la valeur 0 dans le registre TIMBRE.

10.0 LECTURE DES MANETTES DE JEU

* code du point d'entrée: 1CH pour un JSR, 9CH pour un JMP.

* paramètre d'entrée:

- registre 6809 A

* paramàtres de retour:

- registres 6809 B et CC

Le numéro (0 ou 1) de la manette de jeu dont on veut connaître l'état est passé per l'accumulateur A.

B retourne une valeur de 0 à 8 donnant l'état de la manette de jeu, suivant la convention que voici:

0 ==> Centre

1 ==> Nord

2 ==> Nord Est

3 ==> Eat

4 ==> Sud Est

5 ==> Sud

6 ==> Sud Ouest

7 ==> Ouest

8 ==> Nord Ouest

Le bit de retenue du registre CC est mis à 1 si le bouton a été enfoncé, sinon il est mis à zéro.

11.0 CRAYON OPTIQUE

11.1 Test du bouton du crayon optique

* code du point d'entrée: 16H pour un JSR, 96H pour un JMP.

* paramètre de retour:

- registre 6809 CC

Cette routine teste l'état du bouton du crayon optique. Si celui-ci a été enfoncé, le bit de retenue est forcé à 1, sinon il est forcé à 0.

11.2 Lecture de la zone pointée

* code du point d'entrée: 18H pour un JSR, 98H pour un JMP.

* paramètres de retour:

- registres 6809 X,Y et CC

Cette routine lit les coordonnées du point visé par le crayon optique, et retourne l'abscisse dans X (de 0 a 319) et l'ordonnée dens Y (de 0 a 199). Si la mesure est correcte, le bit de retenue du registre CC est forcé à zéro. En cas de mauvaise lecture (luminosité trop faible, crayon ne pointant pas l'écran), ce bit est forcé à 1.

12.0 GESTION DE L'INTERFACE DE COMMUNICATION

* code du point d'entrée: 24H pour un JSR, A4H pour un JMP.

* paramètres d'entrée:

- registre 6809 B

- registres PR.OPC (2042H) et GRCODE (2077H)

* paramètres de retour:

- registre 6809 CC

- registres PR.STA (2043H)

Cette routine gère l'interface de communication. Le contenu du registre sélectionne l'une des opérations suivantes:

CONTENU DE PR.OPC	OPERATION DEMANDEE
% 0 0 0 0 0 0 0 1	Ecriture d'un octet en parallèle.
% 0 0 0 0 0 0 1 0	Copie graphique d'écran.
% 0 0 0 0 0 1 0 0	Ouverture pour écriture en parallèle.
% 0 0 0 1 0 0 0 0	Fermeture.

Dans le cas d'un octet à écrire en perallèle, B doit contenir l'octet à envoyer.

En ces de copie graphique de l'écran, le registre GRCODE doit contenir le code de mise en mode graphique spécifique de l'imprimante. Ce registre contient la valeur 7 par défaut.

Le registre PR.STA retourne le code de l'opération réalisée :

CONTENU DE PR.STA	ETAT DE LA COMMUNICATION
% 0 0 0 0 0 1 0 0	Ouvert pour écriture en perallèle.
% 0 0 0 0 1 0 0 0	Périphérique non prêt
% 0 0 0 1 0 0 0 0	Fermé.

Le bit de retenue du registre CC est forcé à zéro si tout s'est passé normalement, sinon il est forcé à 1.

13.0 GESTION DU LECTEUR-ENREGISTREUR DE PROGRAMMES (LEP)

13.1 Lecture et écriture de la bande

* code du point d'entrée: 20H pour un JSR, A0H pour un JMP.

* paramètrea d'entrée;

- registres 6809 A, B et Y

- registrea K7DATA (2040H) et K7LENG (2041H)

* paramètres de retour:

- registres 6809 A et B.

Cette routine permet de lire ou d'écrire des blocs de 253 octets de données au maximum. Ces blocs peuvent être de trois types, le type étant codé de la facon suivante:

- 00: bloc d'en-tête du fichier.

- 01: bloc de données.

- FFH: bloc de fin de fichier.

Si A est non nul, la routine effectue une lecture de la bande magnétique. Y doit contenir l'adresse de la zone mémoire où sera rangé le bloc lu. Au retour, B contiendra le type de bloc, et A le "checksum" calculé d'après les donnéea lues (somme des données et du checksum lu). A l'adresse pointée par Y, l'on trouvera:

- un octet donnant la longueur du bloc.

- n octets de données.

- un octet donnant le "checksum" lu.

Si A = 0, il s'agit d'une écriture. B doit contenir le type du bloc, et Y doit contenir l'adresse de la zone mémoire où se trouvent lea données à écrire sur la bande. Cette zone doit avoir la structure suivante:

1° octet: longueur du bloc = nombre d'octets de données +2.

2° octet: première donnée.

.

. (Longueur du bloc - 2)données

.

n° octet: (n-1)° donnée.

n+1° octet: checksum de contrôle (opposé de la somme des données si l'on veut en cas de lecture valide avoir un checksum lu qui soit nul).

13.2 Moteur: marche/arrêt

* code du point d'entrée: 22H pour un JSR, A2H pour un JMP.

* paramètre d'entrée:

- registre 6809 A

* paramètre de retour:

- registre 6809 CC

Si le bit 0 de A est à zéro, le moteur est arrêté après une demi-seconde de délai.

Si le bit 0 et le bit 1 de A sont chacun à 1, le moteur est mis en route, et il y a ensuite une seconde de délai.

Si le bit 0 de A est à 1, et le bit 1 de A est à zéro, le moteur est mis en route mais il n'y a pas d'attente ensuite.

La fiabilité sera augmentée si vous suivez les conseils que voici:

* mise en route du moteur pour une écriture: lancement du moteur, puis attente d'une seconde, afin d'être sûr que la vitesse du moteur s'est stabilisée avant d'écrire sur la bande. A doit alors contenir la valeur %xxxxxx11.

* mise en route du moteur pour une lecture: lancement du moteur, pas d'attente. A doit alors contenir la valeur %xxxxxx01.

* arrêt du moteur sprès une lecture ou une écriture: attente d'une demi-seconde pour être sûr qu'aucune partie de la bande contenant des données ne se trouvera sous le cabestan, puis arrêt du moteur. A doit alors contenir ls valeur %xxxxxx10.

Si le LEP n'est pas raccordé à votre MO5, le bit de retenue du registre code condition sera forcé à 1, sinon il sera forcé à zéro.

14.0 CONTROLEUR DE DISQUETTES

Le MO5 est entièrement compatible avec les disquettes TO7, simple ou double densité. Vous pouvez gérer les entrées/sorties disquette à deux niveaux: au niveau physique en utilisant le point d'entrée du moniteur, ou à un niveau logique en manipulant des fichiers au format Basic Microsoft(R).

Rappelons que les disquettes sont divisées en 40 pistes de 16 secteurs chacune. Un secteur contient lui-même 128 octets en simple densité, ou 256 en double densité.

De plus, le contrôleur double deneité peut fonctionner dens les 2 modes simple ou double densité. Au démarrage, sa densité de travail doit être sélectionnée après l'initialisation du contrôleur.

14.1 Entréea/sorties disquette physiques

* code du point d'entrée: 26H pour un JSR, A6H pour un JMP.

* paramètres d'entrée:

- registres DK.OPC (2048H), DK.DRV (2049H), DK.SEC (204CH), DK.TRK (204AH-204BH), DK.8UF (204FH-2050H).

* paramètres de retour:

- registre 6809 CC

- registre DK.STA (204EH)

Le registre DK.OPC contient le code de l'opération à réaliser:

- Code 01: demande l'initialisation du contrôleur. Dans ce cas, si l'initialisetion e pu avoir lieu sans erreurs, le bit de retenue du registre code condition est mis à 0, et le type de contrôleur est retourné dans le registre DK.STA: "C" pour la simple densité, et "D" pour la double densité. Sinon, le bit de retenue est mis à 1, et le code d'erreur 40H est mis dens le registre DK.STA (voir ci-après les codes d'erreur).

- Code 02: lecture d'un secteur. Les codes d'erreur possibles sont 02, 04, 08, 10H, 80H. Si une erreur a eu lieu, son code est retourné dans le registre DK.STA.

- Code 04: sur lecteur double densité, passage du contrôleur en simple densité, pas d'erreur possible. Erreur sur un controleur simple densite.

- Code 08: écriture d'un secteur. Les codes d'erreur possibles sont 01, 02, 04, 08, 10H, 20H, 80H. Si une erreur a eu lieu, son code est retourné dans le registre DK.STA.

- code 10H: sur lecteur double densité, passage du contrôleur en double densité, pas d'erreur possible. Erreur sur un controleur simple densite.

- Code 20H: recherche de la piste zéro. Les codes d'erreur possibles sont 10H et 80H.

- Code 40H: recherche de la piste dont le numéro est passé par le registre DK.TRK. Les codes d'erreur possibles sont 10H et 80H.

- Code 80H: option de vérification. Il feut faire un "OR" logique entre ce code et le code de l'opération que l'on veut vérifier. Les codes d'erreur retournés sont les mêmes que ceux de l'opération spécifiée augmentés du code 20H.

Parsmètres à passer suivant l'opération demandée:

* registre DK.DRV: ce registre doit contenir le numéro du lecteur concerné, soit une valeur entre 0 et 3.

* registre DK.TRK: ce registre doit contenir 39, numéro de piste msximum autorisé, les 40 pistes étant numérotées de 0 à 39.

* registre DK.SEC: ce registre doit contenir le numéro de secteur où l'on veut lire ou écrire. Ce numéro doit être compris entre 1 et 16.

* registre DK.BUF: ce registre doit contenir l'adresse de début d'une zone tampon en RAM de 128 octets en simple densité, ou de 256 octets en double densité, soit pour y lire les données à écrire sur disquette, soit pour y écrire les données lues sur disquette.

* registre DK.STA: ce registre contient le code d'erreur, ou le type de contrôleur pour une initialisation correcte.

Voici les différentes erreurs possibles:

- code 01: disquette protégée; cette erreur ne peut apparaître qu'après une demande d'écriture.

- code 02: erreur de piste; l'identificateur de piste est correct, mais ne correspond pas à la piste demandée.

- code 04: erreur de secteur; l'identificateur de secteur est incorrect (secteur ne pouvant être lu ou erreur sur le checksum), cependent la piste peut être correcte.

- code 08: erreur sur les données; l'identificateur de secteur est correct, mais les données ne peuvent être lues, ou le checksum est incorrect.

- code 10H: lecteur non prêt; le moteur n'est pas en route, ou le lecteur spécifié est inexistant.

- code 20H: erreur sur vérificstion; la zone tsmpon en mémoire et la zone correspondante sur la disquette ne sont pas identiques.

- code 40H: contrôleur non prêt.

- code 80H: disquette non formattée. L'identificeteur de piste ne peut être lu.

14.2 Le format Basic Microsoft(R)

Pour assurer la compatibilité entre les diverses applications, les fichiers créés par le MO5 suivent le standard Basic Microsoft(R).

La piste 20 est une zone réservée destinée à rendre compte de l'état de la disquette. Elle est organisée comme suit:

* secteur 1: réservé

* secteur 2: table d'allocation des fichiers ou "FAT"

* secteurs 3 à 16: catalogue.

14.2.1 La table d'allocation des fichiers

Les fichiers sont organisés en blocs de 1K octets en simple densité, ou 2K octets en double densité. L'on a donc dens tous les cas 2 blocs par piste. Les blocs sont numérotés à partir de 0. Chaque octet de la table d'allocation des fichiers, à partir de l'octet 1, représente un bloc physique.

Organisation de la "FAT":

* Octet 0: 0

* Octet 1: bloc 0, piste 0, secteurs 1 à 8.

* Octet 2: bloc 1, piste 0, secteurs 9 à 16.

* Octet 3: bloc 2, piste 1, secteurs 1 à 8.

* Octet 4: bloc 3, piste 1, secteurs 9 à 16.

* .....

* Octet 2j-1: bloc 2j-2, piste j-1, secteurs 1 à 8.

* Octet 2j: bloc 2j-1, piste j-1, secteurs 9 à 16.

* .....

* Octet 80: bloc 79, piste 39, secteurs 9 à 16.

Un octet de la "FAT" représentant un bloc physique peut avoir comme valeura:

* FFH, qui signifie bloc non alloué.

* FEH, qui signifie bloc réservé.

* Tout nombre de 0 à BFH, signifie bloc alloué. Dans ce cas, le nombre représente le numéro du bloc logique suivant du même fichier.

* Tout nombre de C1H à C8H, signifie dernier bloc d'un fichier. Les 4 bits de poids faible indiquent le nombre de secteurs utilisés dans ce dernier bloc.

14.2.2 Le catalogue

Le catalogue donne la liste des fichiers, et occupe 14 secteurs. Chaque fichier est répertorié sur 32 octets. Il y a donc 4 fichiers répertoriés par secteur en simple densité, et 8 fichiers par secteur en double densité. Au total, le catalogue peut donc répertorier 56 fichiers en simple densité, et 112 en double densité.

Chaque fichier est répertorié de la facon suivante:

Octets 00 à 07: Nom du fichier, cadré à gauche, complété par des blancs.

Octets 08 à 0AH: Suffixe du fichier (.BAS, .BIN, etc...), cadré à gauche, complété par des blancs.

Octet 0BH: Type de fichier: 0 pour un programme Basic ASCII ou binaire, 1 pour des données Basic en ASCII, 2 pour un programme en langage machine (binaire), 3 pour un fichier assembleur édité en ASCII.

Octet 0CH: Sémaphore: FFH pour de l'ASCII, 00 pour du binaire.

Octet 0DH: Numéro du premier bloc logique du fichier.

Octets 0EH-0FH: Nombre d'octets utilisés dans le dernier secteur du fichier.

Octets 10H-1FH: Réservés.

Le premier octet de chaque entrée dans le catalogue indique son état:

* 00: entrée non allouée, pas de fichier répertorié pour cette entrée.

* 20H-7FH: code ASCII du premier caractère du nom de fichier, donc entrée allouée.

* FFH: fin logique du catalogue.

Lors de la création du catalogue, ses octets sont mis à FFH. Chaque fois qu'un fichier est créé, la fin logique du catalogue est déplacée dans le premier octet de l'entrée suivante, jusqu'à ce que le catalogue soit plein. Lorsqu'un fichier est détruit, le premier octet de son entrée est mis à zéro (entrée non allouée). Dans ce cas, tout fichier nouvellement créé se verra attribuer en priorité cette entrée.

15.0 INFORMATIONS COMPLEMENTAIRES

15.1 Organisation de la mémoire

ADRESSES (HEXADECIMAL)	
0000-1FFF	Mémoire écran 2 x 8 K.
2000-20FF	Registres du moniteur.
2100-21FF	Registres de l'application.
2200-9FFF	Mémoire utilisateur env. 32 K
A000-A7BF	1.9 K pour la gestion disquettes.
A7C0-A7C3	PIA 6821 système.
A7C4-A7CB	L I B R E . . .
A7CC-A7CF	PIA 6821 extension jeux.
A7D0-A7DF	Contrôleur de disquettes.
A7E0-A7E3	PIA 6821 interface de communication.
A7E4-A7E7	Compteurs crayon optique.
A7E8-A7FF	Extensions.
A800-AFFF	L I B R E, 2 K.
B000-EFFF	Cartouche ROM, 16 K.
F000-FFFF	Moniteur, 4 K.

15.2 Points d'entrée standard du moniteur

Les codes qui suivent sont les codes sur 7 bits. Le bit 7 doit être mis à zéro pour simuler un JSR, à 1 pour simuler un JMP.

00 : Rend la main à la cartouche assembleur.

02 : Affichage d'un caractère.

04 : Mise en mémoire couleur.

06 : Mise en mémoire forme.

08 : Emission d'un bip sonore.

0AH : Lecture du clavier.

0CH : Lecture rapide du clavier.

0EH : Tracé d'un segment de droite.

10H : Allumage ou extinction d'un point.

12H : Ecriture d'un point "caractères".

14H : Lecture de la couleur d'un point.

16H : Lecture du bouton du crayon optique.

18H : Lecture du crayon optique.

1AH : Lecture de l'écran.

1CH : Lecture des manettes de jeu.

1EH : Génération de musique.

20H : Lecture/écriture sur la cassette.

22H : Mise en route/arrêt du moteur.

24H : Gestion de l'interface de communication.

26H : Contrôleur de disquette.

Les codes de 28H à 5AH (et de A8H à DAH) inclus sont réservés à l'usage interne du moniteur.

15.3 Registres utilisés par le moniteur

Les registres sont donnés par leur nom et leur adresse en HEXADECIMAL.

* 2000-2018 (TERMIN) : Table des terminateurs de lignes logiques.
* 2019 (STATUS) : Différents sémaphores:

b7: 0: majuscule/1: minuscule.
b6: 1: scroll caractère sans couleur.
b5: non utilisé.
b4: 1: graphiques sans écriture de couleur.
b3: sémaphore de lecture clavier.
b2: curseur 1: visible/0: invisible.
b1: sémaphore de répétition clavier.
b0: touche clavier déjà lue.

* 201A-201B (TABPT) : Pointeur dans la table des terminateurs de lignes.
* 201B (RANG) : Ligne logique courante.
* 201C (COLN) : Colonne logique courante.
* 201D-201E (TOPTAB) : Pointeur sur le sommet logique de la table des terminateurs de lignes.
* 201E (TOPRAN) : Première ligne logique de la fenêtre.
* 201F-2020 (BOTTAB) : Pointeur sur la fin logique de la table des terminateurs de ligne.

* 2020 (BOTRAN) : Dernière ligne logique de la fenêtre.
* 2021-2022 (SCRPT) : Pointeur courant dans l'écran.
* 2023-2024 (STADR) : Adresse du premier octet de la fenêtre.
* 2025-2026 (ENDDR) : Adresse + 1 du dernier octet de la fenêtre.
* 2027-2028 (BLOCZ) : Registre de contenu toujours nul, pour les initialisations.
* 2029 (FORME) : Contient le code de la couleur (de -16 à +15) pour la mise en couleur d'un point ou le tracé d'un segment de droite.
* 202A (ATRANG) : Sémaphores pour la gestion d'écrsn:

b7: 1: fond plein écran.
b6: 1: forme plein écran.
b5: réservé.
b4: réservé.
b3: réservé.
b2: réservé.
b1: largeur 0: simple/1: double.
b0: hauteur 0: simple/1: double.

* 202B (COLOUR) : Couleur courante; les 4 bits de poids faible donnent la couleur du fond, les 4 bits de poids fort la couleur de la forme, suivant le codage vidéo pBVR où p est le bit de couleur pastel.
* 202C (PAGFLG) : Si ce registre contient la valeur 0, on est en mode scroll; s'il contient 255, on est en mode "page" (pas de scroll).
* 202D (SCROLS) : Si ce registre est à 0, le scroll est normal; s'il contient 255, le scroll est lent.
* 202E (CURSFL) : Sémaphore de mouvement curseur, qui, s'il contient le valeur 255, indique qu'il ne faut pas lier logiquement le ligne courante à la suivante.
* 202F (COPCHR) : Sémaphore qui, s'il contient la valeur 255, indique que le déplacement à droite ou à gauche doit être interprété comme une recopie à droite ou à geuche du caractère courant.
* 2030 (EFCMPT) : Compteur d'effacement du curseur pour la gestion du clignotement.
* 2031 (ITCMPT) : Compteur d'interruptions IRQ.
* 2032-2033 (PLOTX) : Abscisse du dernier point allumé ou éteint par une routine graphique.
* 2034-2035 (PLOTY) : Ordonnée du dernier point allumé ou éteint par une routine graphique.
* 2036 (CHDRAW) : Code ASCII du caractère pour un tracé de point ou de droite en mode "caractères". Si ce registre contient la valeur 0, le tracé est fait en mode "points".
* 2037 (KEY) : Code de la dernière touche enfoncée au clavier
* 2038 (CMPTKB) : Compteur de répétitions clavier.
* 2039 : Réservé.
* 203A (TEMPO) : Tempo général pour la génération de musique.
* 203B : Réservé.
* 203C (DUREE) : Durée de la note (de 1 à 96).
* 203D (TIMBRE) : Attaque de la note.
* 203E-203F (OCTAVE) : Octave (1, 2, 4, 8 ou 16)
* 2040 (K7DATA) : Octet à écrire sur la bande magnétique du LEP.
* 2041 (K7LENG) : Nombre d'octets consécutifs à écrire sur la bande magnétique du LEP.
* 2042 (PR.OPC) : Mot de commande pour la gestion imprimante.
* 2043 (PR.STA) : Etat courant de la liaison imprimante.
* 2044-2045 (TEMP) : Registre temporaire.
* 2046-2047 (SAVEST) : Sauvegarde du pointeur de pile.
* 2048 (DK.OPC) : Mot de commande pour le contrôleur de disques.
* 2049 (DK.DRV) : Numéro de lecteur de disquettes.
* 204A-204B (DK.TRK) : Numéro de piste.

```
* 204C (DK.SEC) :        Numéro de secteur.
* 204D (DK.NUM) :        Entrelacement de secteurs.
* 204E (DK.STA) :        Etat courant du contrôleur de disquettes.
* 2041 (K7LENG) :        Nombre d'octets consécutifs à écrire sur la
* 204F-2050 (DK.BUF) :   Pointeur sur la zone-tampon réservée aux
                         entrées-sorties disquette.
* 2051-2052 (TRACK0) :   Position de la tête, lecteur 0.
* 2053-2054 (TRACK1) :   Position de la tête, lecteur 1.
* 2055-2056 (TRACK2) :   Position de la tête, lecteur 2.
* 2057-2058 (TRACK3) :   Position de la tête, lecteur 3.
* 2059 (SEQUCE) :        Code indiquant dans quelle séquence de gestion
                         d'écran on se trouve.
* 205A (US1) :           Sémaphore pour les séquences "unit separator".
* 205B (ACCENT) :        Sémaphore pour les séquences accent.
* 205C (SS2GET) :        Registre pour l'impression ou la lecture d'une
                         minuscule accentuée.
* 205D (SS3GET) :        Registre pour l'impression ou la lecture d'une
                         minuscule accentuée.
* 205E-205F (SWI1PT) :   Pointeur sur ls routine de traitement des
                         appels au moniteur.
* 2060 :                 Réservé.
* 2061-2062 (TIMEPT) :   Pointeur sur la routine utilisateur de trai-
                         tement des interruptions IRQ.
* 2063 :                 Sémaphore d'aiguillage des IRQ.
* 2064-2065 (IRQPT) :    Pointeur sur la routine moniteur de traitement
                         des interruptions IRQ.
* 2066 :                 Réservé.
* 2067-2068 (FIRQPT) :   Pointeur sur la routine de traitement des in-
                         terruptions rapides FIRQ.
* 2069 :                 Réservé.
* 206A-206B (SIMUL) :    Pointeur sur la table des points d'entrée du
                         moniteur.
* 206C :                 Réservé.
* 206D-206E (CHRPTR) :   Pointeur sur la table de décodage du clavier.
* 206F :                 Réservé.
* 2070-2071 (USERAF) :   Pointeur sur le générateur de caractères
                         utilisateur.
* 2072 :                 Réservé.
* 2073-2074 (GENPTR) :   Pointeur sur le générateur de caractères stan-
                         dard
* 2075 :                 Réservé.
* 2076 (LATCLV) :        Latence clavier.
* 2077 (GRCODE) :        Mot de code pour la mise en mode graphique de
                         l'imprimante.
* 2078 (DECALG) :        Ajuatement pour le crayon optique.
* 2079-207E :            Réservés., 2 0.
* 207F (DEFDST) :        Sémaphore de simple ou double densité.
* 2080 (DKFLG) :         Sémaphore de présence du contrôleur de disques
* 2081-20CC :            Pile système.
* 20CD-20E4 (LPBUFF) :   Zone-tampon pour la lecture du crayon optique.
* 20E5-20FD :            Réservés.
* 20FE-20FF (TSTRST) :   Sémaphore de démarrage à chaud ou à froid.
```

15.4 Adresses d'entrées/sorties

Les adresses qui suivent sont données en hexadécimal.

15.4.1 PIA système

*** PIA système 6821 ***

```
* A7C0:                 Registre de données, port A.

        bit0 (sortie):  commutation mémoire écran caractères et
                        mémoire écran couleur.
```

```
        bit1 (sortie):  couleur du tour, rouge.
        bit2 (sortie):  couleur du tour, vert.
        bit3 (sortie):  couleur du tour, bleu.
        bit4 (sortie):  couleur du tour, pastel.
        bit5 (entrée):  interruption crayon optique.
        bit6 (sortie):  écriture cassette.
        bit7 (entrée):  lecture cassette.

* A7C1:                 Registre de données, port B:

        bit0 (sortie):  son
        bits 1-6: (sortie): matriçage/dematriçage clavier.
        bit7 (entrée):  lecture clavier.

* A7C2:                 Registre de contrôle, port A:
CA1 (entrée):   crayon optique.
CA2 (sortie):   moteur du LEP.

* A7C3:                 Registre de contrôle, port B:
CB1 (entrée):   interruptions 50 Hertz.
CB2 (sortie):   commande d'incrustation.
```

15.4.2 PIA jeux

```
                    *** PIA JEUX 6821 ***

* A7CC:                 Registre de données, port A:

        bits 0-7 (entrée): lecture des manettes de jeu.

* A7CD:                 Registre de données, port B:

        bits 0-5 (entrée): convertisseur digital/analogique.
        bit6 (entrée):  action manette de jeu 0.
        bit7 (entrée):  action manette de jeu 1.

* A7CE:                 Registre de contrôle, port A:
CA1 (entrée):   sction manette de jeu 0.

* A7CF:                 Registre de contrôle, port B:
CB1 (entrée):   action manette de jeu 1.
```

15.4.3 Interface de communication

```
            *** INTERFACE DE COMMUNICATION 6821 ***

* A7E0:                 Registre de données, port A:
Non utilisé, disponible pour la gestion d'une liaison série.

* A7E1:                 Registre de données, port B:

       Bits 0-7 (sortie) données en parallèle.

* A7E2:                 Registre de contrôle, port A:
CA1 (entrée): request to send (demande d'émission).

* A7E3:                 Registre de contrôle, port B:
    CB1 (entrée): acknowledge.
    CB2 (sortie): strobe.
```

15.5 Profondeur de la pile dans le moniteur

Lors de tout appel au moniteur se faisant par un SWI, il y a au moins 16 octets empilés au cours d'un appel.

Certaines routines du moniteur sont protégées contre les interruptions: c'est le cas des routines de générstion de musique, de lecture/écriture de la csssette, de saisie du crayon optique, et des routines de gestion des disquettes.

Pour les autres, il faut tenir compte de la possibilité d'interruption par IRQ ou FIRQ, ce qui ajoute respectivement 30 et 3 octets à la profondeur de pile nécessaire.

Voici la liste de la profondeur de pile MINIMUM nécessaire selon les codes des différents modules:

Code 02H: de 16 à 43 octets selon les caractères:

Caractère normal affichable : minimum 21 octets, maximum 43 octets si il y a scroll, y compris pour les minuscules sccentuées.

Caractère de contrôle non affichable :

07H	Bip sonore	16 octets
08H	Flèche à gauche	23 octets, 43 si scroll.
09H	Flèche à droite	23 octets, 43 si scroll.
0AH	Flèche en bas	21 octets, 43 si scroll.
0BH	Flèche en haut	21 octets, 43 si scroll.
0CH	RAZ écran	21 octets.
0DH	Retour chariot	21 octets.
11H	Curseur visible	16 octets.
14H	Curseur invisible	21 octets
16H	Séquence accent	16 octets.
18H	"Cancel"	35 octets.
1BH	Echappement	16 octets.
1EH	"RS"	21 octets.
1FH	Séquence US	16 octets.

Les codes non reconnus, ne sont pas traités et provoquent donc seulement l'empilement de 16 octets.

* séquences d'échappement (taille pour chaque appel):

- plein écran : inverse vidéo 35 octets
  couleur fond 38 octets
  couleur forme 38 octets

- courant : inverse vidéo 22 octets
  scroll 18 octets
  incrustation 18 octets
  pas incrust. 18 octets
  écr.car.seul 18 octets
  écr.csr&coul 18 octets
  taille 23 octets
  tour saturé 23 octets
  couleur fond 23 octets
  car. saturé 23 octets

* séquences US :

- code < $40 : 26 octets
- code >= $40 : 25 octets

Code 04H: 16 octets.

Code 06H: 16 octets.

Code 08H: 16 octets.

Code 0AH: 32 octets.

Code 0CH: 16 octets.

Code 0EH: 23 octets si le tracé est horizontal,
32 s'il est graphique,
53 s'il est en caractères.

Code 10H: 18 octets si graphique,
40 si caractère.

Code 12H: 40 octets.

Code 14H: 18 octets.

Code 16H: 17 octets.

Code 18H: 18 octets.

Code 1AH: 36 octets.

Code 1CH: 16 octets.

Code 1EH: 18 octets.

Code 20H: 20 octets en lecture,
18 en écriture.

Code 22H: 16 octets (marche ou arrêt).

Code 24H: 16 octets pour ouverture et écriture carectère,
30 pour une copie graphique d'écran.

Code 26H: 40 octets en lecture ou écriture.

Interruption rapide FIRQ: 3 octets

Interruption normale IRQ: 30 octets

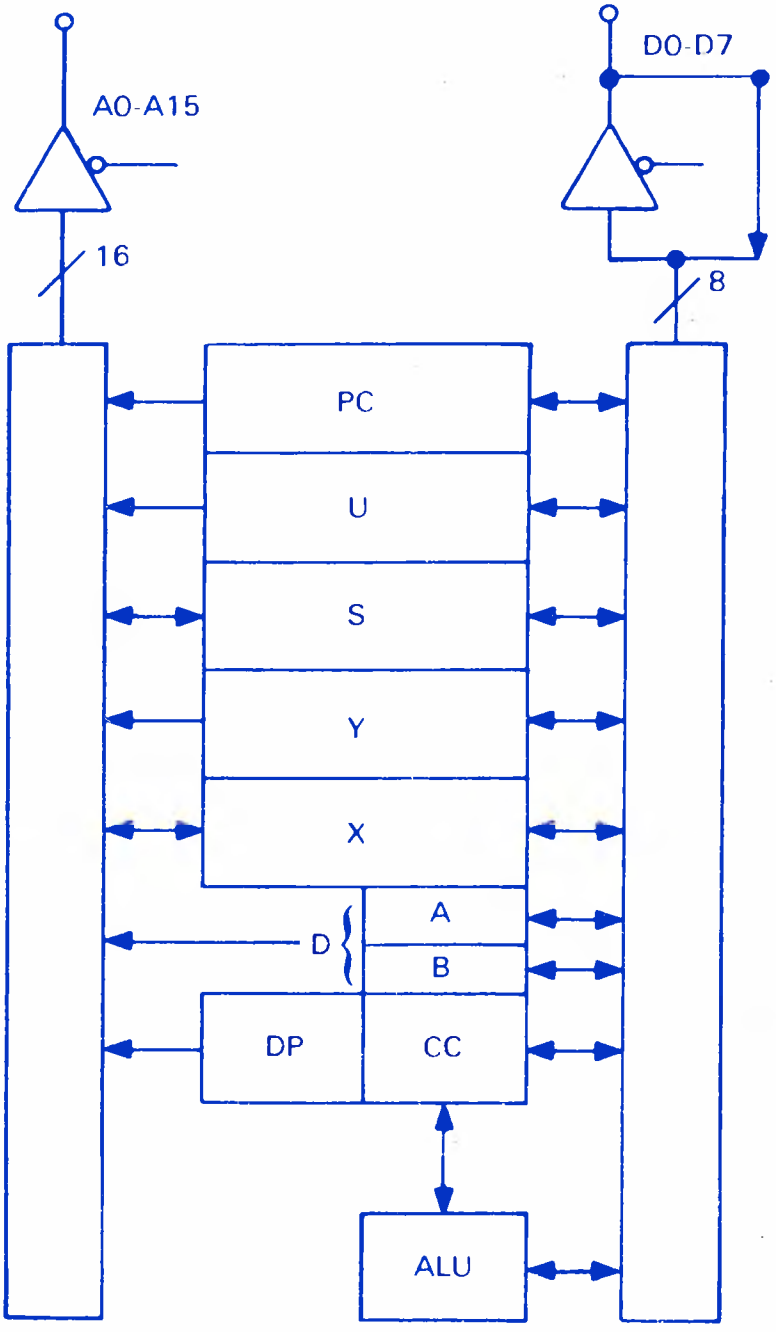
ANNEXE B
MICROPROCESSEUR 6809
INSTRUCTIONS ET ADRESSAGE

Extrait de *Microprocesseurs et périphériques*, édité par Thomson Semiconducteurs, 45, avenue de l'Europe, 78140 Velizy, téléphone (1) 946.97.19, télex 698 866 F .

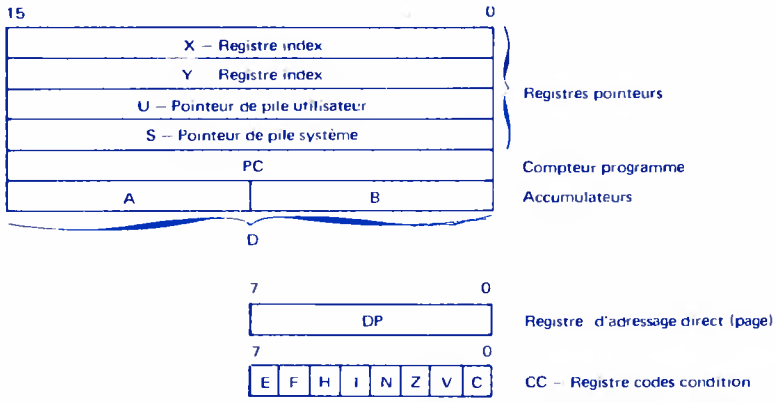
Le circuit 6809 est un microprocesseur 8 bits de conception révolutionnaire, utilisant les techniques de programmation moderne telles que banalisation de l'implantation en mémoire, réentrance et programmation modulaire.
Cet apport de 3e génération à la famille 6800 offre des améliorations d'architecture qui incluent des registres, des instructions et des modes d'adressage supplémentaires.
Les instructions de base de tout ordinateur sont particulièrement améliorées par la présence de modes d'adressage puissants. Le jeu de modes d'adressage disponible du 6809 est actuellement le plus complet des microprocesseurs existants. Les caractéristiques logicielles et matérielles du circuit, en font un processeur idéal pour l'exécution de programmes en langages évolués ou pour la réalisation d'applications standards.

- 10 modes d'adressage
  Compatibilité ascendante des modes d'adressage avec la famille 6800.
  Adressage direct dans tout l'espace mémoire.
  Branchements relatifs longs.
  Compteur programme relatif.
  Indirection.
  Adressage indexé étendu.
    déplacements constants 0, 5, 8, 16 bits
    déplacements accumulateur 8, 16 bits
    auto-incrémentation/décrémentation
- Manipulation de pile améliorée
- 1464 instructions
- Multiplication non signée 8×8 bits
- Arithmétique 16 bits
- Transfert/échange tous registres
- Empilement/dépilement de chacun ou de l'ensemble des registres
- Adresse effective de chargement

SCHÉMA FONCTIONNEL DU 6809



REGISTRES PROGRAMMABLES DU MICROPROCESSEUR



REGISTRES PROGRAMMABLES

Comme indiqué ci-dessus, le microprocesseur 6809 comporte trois registres supplémentaires par rapport au 6800. Ces registres sont les suivants : un registre de page directe (DP), un registre pointeur de pile utilisateur (U) et un second registre index (Y).

ACCUMULATEURS (A, B, D)

Les registres A et B sont des accumulateurs universels utilisés pour les calculs arithmétiques et les manipulations de données.
Certaines instructions concatènent les registres A et B pour former un seul accumulateur 16 bits. Le registre A constitue l'octet de poids fort de cet accumulateur référencé registre D.

REGISTRE PAGE DIRECTE (DP)

Le registre de page directe du circuit 6809 est utilisé pour étendre les possibilités d'adressage en mode direct. Le contenu de ce registre apparaît aux sorties d'adresse de poids forts (A8-A15) pendant l'exécution d'une instruction d'adressage direct. Ce registre permet d'utiliser le mode d'adressage direct, sous le contrôle du programme, dans tout l'espace d'adressage. Pour permettre la compatibilité avec la famille 6800, tous les bits de ce registre sont mis à zéro à l'initialisation du processeur.

REGISTRES INDEX (X, Y)

Les registres d'index sont utilisés pour les modes d'adressage indexé. Lors des calculs d'adresse effective, les 16 bits de ce registre sont utilisés. Les adresses contenues dans ces registres peuvent servir comme

pointeur de données et être modifiées par une constante optionnelle ou par une valeur de déplacement. Lors de rangement de données sous forme de table, dans certains modes d'adressage indexé, le contenu des registres d'index est incrémenté ou décrémenté pour pointer sur l'élément suivant. Les quatre registres (X, Y, U, S) peuvent être utilisés comme des registres d'index.

POINTEURS DE PILE (U,S)

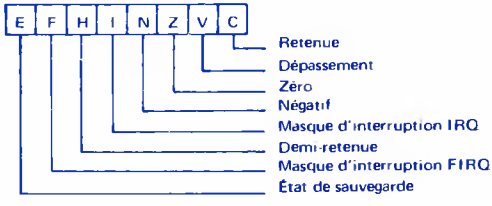
Le pointeur de pile (S) est utilisé automatiquement par le processeur pour mémoriser les états de la machine pendant l'exécution de sous-programmes et interruptions. Les pointeurs du 6809 pointent le haut de la pile, à l'opposé du pointeur du 6800, qui pointait l'emplacement libre suivant sur la pile. Le pointeur de pile utilisateur (U) est commandé par le programmeur exclusivement, permettant ainsi le passage de paramètres de et vers des sous-programmes avec facilité. Les pointeurs de pile U et S ont les mêmes possibilités que les registres X et Y pour les modes d'adressage indexé et pour les instructions d'empilement/dépilement. Le micro processeur 6809 peut être utilisé comme processeur avec gestion de pile, autorisant ainsi l'utilisation de langage de haut niveau et des méthodes de programmation modulaire.

COMPTEUR PROGRAMME (PC)

Le compteur programme est utilisé par le processeur pour pointer l'adresse de l'instruction suivante devant être exécutée. L'adressage relatif permet au compteur programme d'être utilisé comme un registre index dans certains cas.

REGISTRE DE CODES CONDITION (CC)

Le registre de codes condition définit l'état du processeur à tout instant.



DESCRIPTION DU REGISTRE CODES CONDITION (CC)

BIT 0 (C)
Le bit 0 est l'indicateur de retenue, il indique généralement la retenue lors d'une opération de l'unité arithmétique et logique. C est aussi utilisé pour représenter la retenue lors d'instructions correspondant à une soustraction (CMP, NEG, SUB, SBC). Dans ce cas cet indicateur est le complément de la retenue lors d'une opération de l'unité logique et arithmétique.

BIT 1 (V)
Le bit 1 est l'indicateur de débordement ; il est mis à un s'il y a débordement, en complément à deux signé après une opération arithmétique. Le débordement est détecté lors d'une opération dans l'unité logique et arithmétique quand la retenue du MSB ne correspond pas à la retenue du MSB-1.

BIT 2 (Z)
Le bit 2 est le bit indicateur de zéro, il est mis à un si le résultat de l'opération précédente est nul.

BIT 3 (N)
Le bit 3 indique un résultat négatif, il contient exactement la valeur du bit de poids fort de l'octet résultant de l'opération précédente. Un résultat négatif, en complément à deux, positionne N à 1.

BIT 4 (I)
Le bit 4 est le bit masque des interruptions IRQ. Ce bit mis à un, le processeur ne prendra pas en compte les interruptions arrivant sur la ligne IRQ. NMI, FIRQ, IRQ, RESET et SWI positionnent toutes 1 à un. SWI 2 et SWI 3 n'affectent pas 1.

BIT 5 (H)
Le bit 5 est le bit de demi-retenue ; il est utilisé pour indiquer une retenue du bit 3 dans l'ALU comme résultat d'une addition 8 bits seulement (ADC ou ADD). Ce bit est utilisé dans une instruction DAA pour réaliser une opération d'ajustement décimal. L'état de cet indicateur est indéfini dans toutes les instructions de soustraction ou équivalentes.

BIT 6 (F)
Le bit 6 est le bit masque des interruptions rapides FIRQ. Le processeur ne prendra pas en compte les interruptions de la ligne FIRQ, lorsque ce bit est à un. NMI, FIRQ, SWI, et RESET positionnent toutes F à un. F n'est pas affecté par IRQ, SWI 2 et SWI 3.

BIT 7 (E)
Le bit E est le bit indicateur de l'état de sauvegarde ; mis à un, il indique que l'état complet de la machine (tous les registres) est empilé, à la place de l'état précédent (PC et CC). Le bit E du registre CC empilé est utilisé sur un retour d'interruption (RTI) pour déterminer l'étendue du dépilement. Par conséquent, le bit E courant laissé dans le registre CC représente l'action précédente.

MODES D'ADRESSAGE

Les instructions de base de tout ordinateur sont particulièrement améliorées par la présence de modes d'adressage puissants. Le jeu de modes d'adressage disponible du 6809 est actuellement le plus puissant des microprocesseurs existants.
Par exemple, le 6809 possède 59 instructions de base, mais il admet 1464 possibilités différentes d'instructions et de modes d'adressage. Les nouveaux modes d'adressage permettent les techniques de programmation modernes. Les modes d'adressage suivants sont disponibles dans le 6809 :

Inhérent ou implicite (Inclut les accumulateurs)
Immédiat
Etendu
Etendu indirect
Direct
Registre
Indexé
- **Déplacement nul**
- **Déplacement constant**
- **Déplacement accumulateur**
- **Auto incrémentation/décrémentation**
- **Indexé indirect**

Relatif
- **Branchement relatif long/court**
- **Adressage relatif compteur programme**

INHÉRENT (INCLUT LES ACCUMULATEURS)

Dans ce mode d'adressage, le code opération de l'instruction contient toute l'information adresse nécessaire. Des exemples d'adressage inhérent sont : ABX, DAA, SWI, ASRA, CLRB etc.

ADRESSAGE IMMÉDIAT

En adressage immédiat, l'adresse effective des données se trouve à l'emplacement suivant immédiatement le code opération ; les données à utiliser comme adresse dans l'instruction du 6809 utilisent deux valeurs immédiates 8 et 16 bits en fonction de la taille de l'opérande spécifié par le code opération. Des exemples d'instructions utilisant l'adressage immédiat sont :

```
LDA     #$20
LDX     #$F000
LDY     #ASTER
```

Note : # signifie adressage immédiat, $ signifie valeur hexadécimale.

ADRESSAGE ÉTENDU

En adressage étendu, le contenu des deux octets suivant immédiatement le code opération spécifie complètement l'adresse 16 bits effective utilisée par l'instruction.

Il est à noter que l'adresse générée par une instruction étendue définit une adresse absolue et n'est pas translatable. Les exemples d'adressage étendu incluent :

```
LDA     ASTER
STX     OBEL
LDD     $2000
```

ÉTENDU INDIRECT

Comme cas spécial d'adressage indexé (exposé ci-dessous), un niveau d'indirection peut être ajouté à l'adressage étendu. En mode étendu indirect, les deux octets suivant le post octet d'une instruction indexée contiennent l'adresse de l'adresse des données.

```
LDA     [ASTER]
LDX     [$ FFFE]
STU     [OBEL]
```

ADRESSAGE DIRECT

L'adressage direct est similaire à l'adressage étendu excepté qu'un octet d'adresse seulement suit le code opération. Cet octet spécifie les 8 bits de poids faible de l'adresse à utiliser. Les 8 bits d'adresse de poids fort sont fournis par le registre page directe. Un octet d'adresse étant seulement nécessaire en adressage direct, ce mode nécessite moins de mémoire et s'exécute plus rapidement qu'en adressage étendu. Bien entendu, seuls 256 emplacements (une page) peuvent être définis sans avoir à repositionner le contenu du registre DP. Le registre DP étant mis à $ 00 à l'initialisation, l'adressage direct sur le 6809 est compatible avec l'adressage direct du 6800. L'indirection n'est pas permise en adressage direct.

Voici quelques exemples d'adressage direct :

```
LDA     $30
SETDP   $10 (directive d'assemblage)
LDB     $1030
LDD     <ASTER
```

Note : < est une directive d'assemblage qui force l'adressage direct.

ADRESSAGE PAR REGISTRE

Certains codes opération sont suivis d'un octet qui définit un registre ou un jeu de registres devant être utilisés par l'instruction, cet octet est appelé POST OCTET.

Quelques exemples d'adressage registre sont :

```
TFR     X,Y       Transfert de X dans Y
EXG     A,B,      Échange A et B
PSHS    A,B,X,Y   Transfert dans S  Y, X, B puis A
PULU    X, Y, D   Transfert depuis U  D, X, puis Y
```

ADRESSAGE INDEXÉ

Dans tout adressage indexé, un des registres pointeur (X, Y, U, S et parfois PC) est utilisé dans le calcul de l'adresse effective de l'opérande devant être utilisée par l'instruction. Cinq types d'indexation de base sont disponibles et sont exposés ci-dessous. Le post octet d'une instruction indexée spécifie le type de base et le choix du mode d'adressage ainsi que le registre pointeur devant être utilisé. Le tableau ci-dessous montre les formats autorisés pour le post octet. Le tableau "MODES d'Adressage INDEXÉ" donne la forme assembleur et le nombre de cycles et d'octets additionnés aux valeurs de base d'adressage indexé pour chaque variante.

Signification des bits du registre post-octet dans l'adressage indexé

Bit du registre post-octet								Mode d'adressage indexé
7	6	5	4	3	2	1	0	
0	R	R	X	X	X	X	X	EA = ,R ± 4 bits déplacement
1	R	R	0	0	0	0	0	,R+
1	R	R	I	0	0	0	1	,R++
1	R	R	0	0	0	1	0	, -R
1	R	R	I	0	0	1	1	, --R
1	R	R	I	0	1	0	0	EA= ,R ± 0 déplacement
1	R	R	I	0	1	0	1	EA= ,R ± ACCB déplacement
1	R	R	I	0	1	1	0	EA= ,R ± ACCA déplacement
1	R	R	I	1	0	0	0	EA = ,R ± 7 bits déplacement
1	R	R	I	1	0	0	1	EA= ,R±15 bits déplacement
1	R	R	I	1	0	1	1	EA= ,R ± D déplacement
1	X	X	I	1	1	0	0	EA= ,PC±7 bits déplacement
1	X	X	I	1	1	0	1	EA= ,PC±15 bits déplacement
1	R	R	1	1	1	1	1	EA = , adresse

Champ du mode d'adressage

Champ indirect
bit de signe quand B7= 0

Champ du registre
00 :R = X
01 :R = Y
10 :R = U
11 :R = S
X = indifférent

Indexé — Déplacement zéro. Dans ce mode, le registre pointeur sélectionné contient l'adresse effective des données devant être utilisées par l'instruction. Ce mode est le mode indexé le plus rapide. Exemples :

```
LDD     0,X
LDA     0,S
```

Indexé — Déplacement constant. Dans ce mode, un déplacement en complément à deux et le contenu d'un des registres pointeurs sont additionnés pour former l'adresse effective de l'opérande. Le contenu initial du registre pointeur n'est pas changé par l'addition. Trois valeurs de déplacement sont disponibles.

± 4-bit(− 16 à + 15)
± 7-bit(− 128 à + 127)
± 15-bit(− 32 768 à + 32 767)

Le déplacement 5 bits en complément à deux est compris dans le post octet et donc optimise l'utilisation des octets et des cycles. Le déplacement 8 bits en complément à deux est contenu dans un seul octet suivant le post octet. Le déplacement 16 bits en complément à

deux se trouve dans les deux octets suivant le post octet. Dans la plupart des cas, le programmateur n'a pas à connaître la valeur de ce déplacement puisque l'assembleur sélectionne automatiquement la valeur d'option.
Exemples d'indexation avec déplacement constant :

```
LDA     23,X
LDX     -2,S
LDY     300,X
LDU     ASTER,Y
```

Mode d'adressage indexé : non indirect

Type	Formes	Non Indirect			
		Syntaxe assembleur	Post-octet code OP	+	~ #
Déplacement constant à partir de R (signé)	pas de déplacement	,R	1RR00100	0	0
	déplacement 5 bits	n, R	0RRnnnnn	1	0
	déplacement 8 bits	n, R	1RR01000	1	1
	déplacement 16 bits	n, R	1RR01001	4	2
Accumulateur utilisé comme déplacement pour le Registre R (déplacement signé)	registre de déplac. A	A, R	1RR00110	1	0
	registre de déplac. B	B, R	1RR00101	1	0
	registre de déplac. D	D, R	1RR01011	4	0
Auto incrémentation/décrémentation du registre R	incrémenté par 1	,R+	1RR00000	2	0
	incrémenté par 2	,R++	1RR00001	3	0
	décrémenté par 1	,-R	1RR00010	2	0
	décrémenté par 2	,--R	1RR00011	3	0
Déplacement constant à partir de PC	déplacement 8 bits	n, PCR	1XX01100	1	1
	déplacement 16 bits	n, PCR	1XX01101	5	2
Indirect étendu	adresses 16 bits	—	—	—	—

R = X, Y, U ou S — X = 00 Y = 01
X = indifférent — U = 10 S = 11

Mode d'adressage indexé : indirect

Type	Formes	Indirect			
		Syntaxe assembleur	Post-octet code OP	+	~ #
Déplacement constant à partir de R (signé)	pas de déplacement	[,R]	1RR10100	3	0
	déplacement 5 bits	par défaut – 8 bits			
	déplacement 8 bits	[n, R]	1RR11000	4	1
	déplacement 16 bits	[n, R]	1RR11001	7	2
Accumulateur utilisé comme déplacement pour le Registre R (déplacement signé)	registre de déplac. A	[A, R]	1RR10110	4	0
	registre de déplac. B	[B, R]	1RR10101	4	0
	registre de déplac. D	[D, R]	1RR11011	7	0
Auto incrémentation/décrémentation du registre R	incrémenté par 1	impossible			
	incrémenté par 2	[,R++]	1RR10001	6	0
	décrémenté par 1	impossible			
	décrémenté par 2	[,--R]	1RR10011	6	0
Déplacement constant à partir de PC	déplacement 8 bits	[n, PCR]	1XX11100	4	1
	déplacement 16 bits	[n, PCR]	1XX11101	8	2
Indirect étendu	adresses 16 bits	[n]	10011111	5	2

R = X, Y, U ou S — X = 00 Y = 01
X = indifférent — U = 10 S = 11

+ et ~ indiquent le nombre de cycles et d'octets additionnels pour un état particulier.

Indexé — Déplacement accumulateur. Ce mode est semblable au mode indexé à déplacement constant, excepté que la valeur en complément à deux dans un des accumulateurs (A, B ou D) et le contenu de l'un des registres pointeurs sont ajoutés pour former l'adresse effective de l'opérande. Le contenu du registre pointeur et de l'accumulateur demeure inchangé par l'addition. Le post octet spécifie l'accumulateur à utiliser comme déplacement et aucun octet supplémentaire n'est nécessaire. L'avantage d'un déplacement accumulateur réside dans le fait que la valeur du déplacement peut être calculée par programme en cours d'exécution.
Exemples :

```
LDA     B, Y
LDX     D, Y
LEAX    B, X
```

Indexé — Auto Incrémentation/Décrémentation. En mode auto incrémentation, le registre pointeur contient l'adresse de l'opérande. Ainsi, après avoir été utilisé le registre pointeur est incrémenté de un ou deux. Ce mode d'adressage est très utile lors de l'utilisation de tables, de déplacement de données, ou pour la création de piles logicielles. En auto décrémentation le registre pointeur est décrémenté avant d'être utilisé comme adresse des données. L'utilisation en auto décrémentation est similaire à celle en auto incrémentation, mais les tables sont scrutées les adresses élevées vers les adresses faibles. La valeur d'incrément/décrément peut être égale à un ou deux pour permettre d'accéder à des tables de données 8 ou 16 bits, elle est sélectionnée par le programmeur. L'aspect pré-décrément, post-incrément permet à ces modes d'être utilisés pour créer des piles logicielles supplémentaires qui se comportent de manière identique aux piles U et S.
Voici quelques exemples de modes d'adressage auto incrément/décrément :

```
LDA     ,X+
STD     ,Y++
LDB     ,-Y
LDX     ,--X
```

INDEXE INDIRECT
Tous les modes indexé indirect sont inclus à l'exception d'incrémentation/décrémentation par un, ou déplacement de ±4 bits et peu-

vent avoir un niveau d'indirection supplémentaire spécifié. En adressage indirect, l'adresse effective est contenue à l'emplacement spécifié par le contenu du registre index, additionné d'un quelconque déplacement. Dans l'exemple ci-dessous, l'accumulateur A est chargé indirectement en utilisant une adresse effective calculée à partir du registre index et d'un déplacement.

```
Avant exécution :
A=x x(indifférent)
X=$  F000
$0100 LDA [10,X] l'EA est alors $F010
$F010 $F1        F150 est alors la nouvelle
$F011 $50        adresse effective
$F150 $AA
Après exécution :
A=$AA Donnée chargée
```

Note : EA = adresse effective.
Tous les modes indexé indirect sont inclus à l'exception de ceux qui sont sans signification (exemple : auto incrément/décrément par 1 indirect). Quelques exemples de mode indexé indirect sont :

```
LDA     [,X]
LDD     [10,X]
LDA     [B,Y]
LDD     [,X+ +]
```

ADRESSAGE RELATIF

Le(s) octet(s) suivant(s) le code opération de branchement est (sont) traité(s) comme un déplacement signé qui est additionné au compteur programme.
Si la condition de branchement est vraie, alors l'adresse calculée (PC + déplacement signé) est chargée dans le compteur programme. L'exécution du programme se poursuit jusqu'au nouvel emplacement comme indiqué par le PC, les modes d'adressage relatif court (1 octet de déplacement) et long (déplacement de deux octets) sont disponibles. Tout emplacement mémoire peut être atteint en mode d'adressage relatif long, l'adresse effective étant interprétée modulo 2^{16}. Quelques exemples d'adressage relatif sont :

```
        BEQ     ASTER   (court)
        BGT     OBEL    (court)
ASTER   LBEQ    BAMBI   (long)
OBEL    LBGT    BUNNY   (long)
```

```
BAMBI       NOP
BUNNY       NOP
```

Le compteur programme peut être utilisé comme registre pointeur avec des déplacements signés de 8 ou 16 bits. Comme en adressage relatif le déplacement est additionné au PC en cours pour former l'adresse effective. L'adresse effective est alors utilisée comme adresse opérande ou adresse données. L'adressage relatif par compteur programme est utilisé pour écrire des programmes translatables. Les tables relatives à un programme particulier gardent la même liaison après translation du programme, si celles-ci sont référenciées en relatif par rapport au compteur programme.

Exemples :

```
LDA         BUNNY, PCR
LEAX        TABLE, PCR
```

Le mode compteur programme relatif étant un type d'indexation, un niveau supplémentaire d'indirection est utilisable.

```
LDA         [ASTER,PCR]
LDU         [OBEL,PCR]
```

JEU D'INSTRUCTIONS DU 6809

Le jeu d'instruction du 6809 est comparable à celui du 6800 et compatible ascendant au niveau du code source. Le nombre de codes opération a été réduit de 72 à 59, mais grâce à son architecture améliorée et modes d'adressage supplémentaires, le nombre de codes opération disponibles (avec les différents modes d'adressage) est passé de 197 à 1 464.

Certaines instructions et certains modes d'adressage sont décrits en détail ci-dessous :

PSHU/PSHS

Ces instructions ont la propriété d'empiler tout(s) registre(s) du MPU soit sur la pile matériel (S) soit sur la pile utilisateur (U) en une seule instruction.

PULU/PULS

Les instructions de dépilement ont la même propriété que les instructions d'empilement, dans l'ordre inverse. L'octet immédiat suivant

le code opération des instructions d'empilement ou de dépilement détermine quel ou quels registres doivent être empilés ou dépilés. La séquence effective d'empilement/dépilement est fixée ; chaque bit détermine un registre unique à empiler/dépiler comme indiqué.

POST OCTET D'EMPILEMENT/DÉPILEMENT

← ordre d'empilement ordre de dépilement →

PC	U	Y	X	DP	B	A	CC	PSHS/PULS

FFFF... ←adresse mémoire croissante... 0000

PC	S	Y	X	DP	B	A	CC	PSHU/PULU

TFR/EXG

Dans le 6809 chaque registre peut être transféré ou échangé avec un autre registre de même format, c'est-à-dire 8 bits à 8 bits ou 16 bits à 16 bits. Les bits 4-7 du post octet définissent le registre source, tandis que les bits 0-3 représentent le registre destination.
Ceci se représente comme suit :

0000 – D	0101 – PC
0001 – X	1000 – A
0010 – Y	1001 – B
0011 – U	1010 – CC
0100 – S	1011 – DP

Note : Toutes les autres combinaisons sont indéfinies et non valables.

Chargement d'adresse effective (LEA)

L'instruction LEA s'exécute en calculant l'adresse effective utilisée dans une instruction indexée et mémorise cette valeur d'adresse, au lieu des données de cette adresse, dans un registre pointeur. Ceci met l'ensemble des caractéristiques d'adressage interne matériel à la disposition du programmeur. Quelques-unes des implications de cette instruction sont illustrées à l'aide d'exemples.
L'instruction LEA permet aussi à l'utilisateur d'accéder à des données quel que soit l'emplacement. Par exemple :

```
        LEAX    MSG1,PCR
        LBSR    PDATA (programme d'impression message)

MSG1    FCC     /MESSAGE/
```

Cet ensemble de programme imprime "message". En écrivant MSG1, PCR, l'assembleur calcule la distance entre l'adresse présente et MSG1. Ce résultat est placé comme une constante dans l'instruction LEAX qui est indexée par la valeur du PC au moment de l'exécution. Peu importe la position du code pendant son exécution puisque le déplacement calculé depuis le PC mettra l'adresse absolue de MSG1 dans le registre pointeur X. Ce code est totalement translatable.

EXEMPLES D'UTILISATION DE L'INSTRUCTION LEA

Instruction		Opération		Commentaire
LEAX	10, X	X + 10	→X	Addition constante sur 5 bits de 10 dans X
LEAX	500, X	X + 500	→X	Addition constante sur 16 bits de 500 dans X
LEAY	A, Y	Y + A	→Y	Addition de l'accumulateur sur 8 bits dans Y
LEAY	D, Y	Y + D	→Y	Addition de l'accumulateur D sur 16 bits dans Y
LEAU	-10, U	U - 10	→U	Soustraction de 10 dans U
LEAS	-10, S	S - 10	→S	Réservation d'une zone dans la pile
LEAS	10, S	S + 10	→S	Remise en ordre de la pile
LEAX	5, S	S + 5	→X	Transfert aussi bien qu'addition

MUL

Multiple les nombres binaires non signés des accumulateurs A et B et place le résultat non signé dans l'accumulateur 16 bits D.

BRANCHEMENTS RELATIFS LONG ET COURT

Le 6809 a la possibilité de réaliser des branchements relatifs au compteur programme sur tout l'espace mémoire. Dans ce mode, en cas de branchement, le déplacement signé de 8 ou 16 bits est additionné à la valeur du compteur programme utilisé comme adresse effective. Ceci permet le branchement du programme n'importe où dans les 64 K d'espace mémoire. Le code translatable peut être facilement généré par l'utilisation du branchement relatif. Les deux branchements court (8 bits) et long (16 bits) sont disponibles.

TABLEAU DE JEU D'INSTRUCTIONS DU 6809

Les instructions du 6809 ont été séparées en cinq catégories différentes qui sont :

- Fonctionnement 8 bits
- Fonctionnement 16 bits
- Instructions portant sur le registre index/pointeur de pile
- Branchements relatifs (long et court)
- Instructions diverses
- Instructions sur valeur hexadécimale

INSTRUCTIONS SUR LES ACCUMULATEURS ET LA MÉMOIRE (8 BITS)

Mnemoniques	Opérations
ADCA, ADCB	Addition du contenu mémoire à l'accumulateur, avec retenue
ADDA, ADDB	Addition du contenu mémoire à l'accumulateur
ANDA, ANDB	ET logique entre mémoire et l'accumulateur
ASL, ASLA, ASLB	Décalage arithmétique à gauche du contenu mémoire ou accumulateur
ASR, ASRA, ASRB	Décalage arithmétique à droite du contenu mémoire ou accumulateur
BITA, BITB	Test de bit mémoire avec l'accumulateur
CLR, CLRA, CLRB	Mise à zéro du contenu de l'accumulateur ou de la mémoire
CMPA, CMPB	Comparaison du contenu mémoire avec l'accumulateur
COM, COMA, COMB	Complément à deux de l'accumulateur ou du contenu mémoire
DAA	Ajustement décimal de l'accumulateur A
DEC, DECA, DECB	Décrémentetion du contenu mémoire ou de l'accumulateur
EORA, EORB	« OU » exclusif du contenu mémoire avec l'accumulateur
EXG R1, R2	Echange de R_1 avec R_2 (R_1, R_2= A, B, CC, DP)
INC, INCA, INCB	Incrémentation du contenu mémoire ou de l'accumulateur
LDA, LDB	Chargement de l'accumulateur avec le contenu mémoire
LSL, LSLA, LSLB	Décalage logique à gauche du contenu mémoire ou de l'accumulateur
LSR, LSRA, LSRB	Décalage logique à droite du contenu mémoire ou de l'accumulateur
MUL	Multiplication non signée (A x B → D)
NEG, NEGA, NEGB	Complément à un du contenu mémoire ou de l'accumulateur
ORA, ORB	« OU » logique mémoire et accumulateur
ROL, ROLA, ROLB	Décalage circulaire à gauche du contenu mémoire ou de l'accumulateur
ROR, RORA, RORB	Décalage circulaire à droite du contenu mémoire ou de l'accumulateur
SBCA, SBCB	Soustraction du contenu mémoire de l'accumulateur
STA, STB	Mise en mémoire du contenu de l'accumulateur
SUBA, SUBB	Soustraction du contenu mémoire de l'accumulateur
TST, TSTA, TSTB	Test mémoire ou accumulateur
TFR, R1, R2	Transfert de R_1 à R_2 (R_1, R_2= A, B, CC, DP)

NOTE : A, B, CC ou DP peuvent être empilés sur (ou dépilés de) chaque pile avec les instructions PSHS, PSHU, (PULS, PULU).

INSTRUCTIONS SUR LES ACCUMULATEURS ET LA MÉMOIRE (16 BITS)

Mnemoniques	Opérations
ADDD	Addition du contenu mémoire à l'accumulateur D
CMPD	Comparaison du contenu mémoire avec l'accumulateur D
EXG D, R	Échange de D avec X, Y, S, U ou PC
LDD	Chargement de l'accumulateur D avec le contenu mémoire
SEX	Extension de signe de l'accumulateur B à l'accumulateur A.
STD	Mise en mémoire de l'accumulateur D.
SUBD	Soustraction du contenu mémoire de l'accumulateur D
TFR D, R	Transfert de D vers X, Y, S, U ou PC
TFR R, D	Transfert de X, Y, S, U ou PC vers D.

INSTRUCTIONS SUR LES REGISTRES INDEX ET LE POINTEUR DE PILE

Mnemoniques	Opérations
CMPS CMPU	Comparaison mémoire avec pointeur de pile
CMPX CMPY	Comparaison mémoire avec registre index
EXG R1 R2	Echange de D, X, Y, S, U ou PC avec D, X, Y, S, U, ou PC
LEAS LEAU	Chargement de l'adresse effective dans le pointeur de pile
LEAX LEAY	Chargement de l'adresse effective dans le registre index
LDS LDU	Chargement du pointeur de pile avec le contenu mémoire
LDX LDY	Chargement du registre index avec le contenu mémoire
PSHS	Empilement de tout (s) registre (s) (sauf S) sur la pile S
PSHU	Empilement de tout (s) registre (s) (sauf U) sur la pile U
PULS	Depilement de tout (s) registre (s) (sauf S) de la pile S
PULU	Dépilement de tout (s) registre (s) (sauf U) de la pile U
STS STU	Mise en mémoire du pointeur de pile
STX STY	Mise en mémoire du registre index
TFR R1 R2	Transfert de D, X, Y, S, U ou PC vers D, X, Y, S, U ou PC
ABX	Addition de l'accumulateur B à X (non signé)

INSTRUCTIONS DE BRANCHEMENT

Mnemoniques	Opérations
BCC LBCC	Branchement si pas de retenue
BCS LBCS	Branchement si retenue
BEQ LBEQ	Branchement si egal
BGE LBGE	Branchement si supérieur ou égal (signé)
BGT LBGT	Branchement si supérieur (signé)
BHI LBHI	Branchement si supérieur (non signé)
BHS LBHS	Branchement si supérieur ou égal (non signé)
BLE LBLE	Branchement si inférieur ou égal (signé)
BLO LBLO	Branchement si inférieur (non signé)
BLS LBLS	Branchement si inférieur ou égal (non signé)
BLT LBLT	Branchement si inférieur (signé)
BMI LBMI	Branchement si négatif
BNE LBNE	Branchement si non égal
BPL LBPL	Branchement si positif
BRA LBRA	Branchement inconditionnel
BRN LBRN	Non branchement
BSR LBSR	Branchement à un sous-programme
BVC LBVC	Branchement si pas de débordement = 0
BVS LBVS	Branchement si débordement = 1

INSTRUCTIONS SPECIALES

Mnemoniques	Opérations
ANDCC	« ET » logique avec le registre codes condition
CWAI	« ET » logique avec le registre codes condition puis attente d'interruption
NOP	Non operation
ORCC	« OU » logique avec le registre codes condition
JMP	Saut inconditionnel
JSR	Saut à un sous-programme
RTI	Retour d'interruption
RTS	Retour de sous-programme
SWI SWI2 SWI3	Interruption programmee
SYNC	Synchronisation avec la ligne d'interruption

VALEURS HEXADECIMALES DU CODE MACHINE

Code OP Mnémonique	Mode		#
00 NEG	Direct	6	2
01 *			
02 *			
03 COM		6	2
04 LSR		6	2
05 *			
06 ROR		6	2
07 ASR		6	2
08 ASL/LSL		6	2
09 ROL		6	2
0A DEC		6	2
0B *			
0C INC		6	2
0D TST		6	2
0E JMP		3	2
0F CLR	Direct	6	2
10 Page 2	—	—	—
11 Page 3	—	—	—
12 NOP	Implicite	2	1
13 SYNC	Implicite	2	1
14 *			
15 *			
16 LBRA	Relatif	5	3
17 LBSR	Relatif	9	3
18 *			
19 DAA	Implicite	2	1
1A ORCC	Immédiat	3	2
1B *			
1C ANDCC	Immédiat	3	2
1D SEX	Implicite	2	1
1E EXG		8	2
1F TFR	Implicite	6	2
20 BRA	Relatif	3	2
21 BRN		3	2
22 BHI		3	2
23 BLS		3	2
24 BHS/BCC		3	2
25 BLO/BCS		3	2
26 BNE		3	2
27 BEQ		3	2
28 BVC		3	2
29 BVS		3	2
2A BPL		3	2
2B BMI		3	2
2C BGE		3	2
2D BLT		3	2
2E BGT		3	2
2F BLE	Relatif	3	2

Code OP Mnémonique	Mode		#
30 LEAX	Indexé	4*	2*
31 LEAY		4*	2*
32 LEAS		4*	2*
33 LEAU	Indexe	4*	2*
34 PSHS	Implicite	5*	2
35 PULS		5*	2
36 PSHU		5*	2
37 PULU		5*	2
38 *			
39 RTS		5	1
3A ABX		3	1
3B RTI		6/15	1
3C CWAI		20	2
3D MUL		11	1
3E *			
3F SWI	Implicite	19	1
40 NEGA	Implicite	2	1
41 *			
42 *			
43 COMA		2	1
44 LSRA		2	1
45 *			
46 RORA		2	1
47 ASRA		2	1
48 ASLA/LSLA		2	1
49 ROLA		2	1
4A DECA		2	1
4B *			
4C INCA		2	1
4D TSTA		2	1
4E *			
4F CLRA	Implicite	2	1
50 NEGB	Implicite	2	1
51 *			
52 *			
53 COMB		2	1
54 LSRB		2	1
55 *			
56 *			
56 RORB		2	1
57 ASRB		2	1
5B ASLB/LSLB		2	1
59 ROLB		2	1
5A DECB		2	1
5B *			
5C INCB		2	1
5D TSTB		2	1
5E *			
5F CLRB	Implicite	2	1

Code OP Mnémonique	Mode		#
60 NEG	Indexé	6*	2*
61 *			
62 *			
63 COM		6*	2*
64 LSR		6*	2*
65 *			
66 ROR		6*	2*
67 ASR		6*	2*
68 ASL/LSL		6*	2*
69 ROL		6*	2*
6A DEC		6*	2*
6B *			
6C INC		6*	2*
6D TST		6*	2*
6E JMP		3*	2*
6F CLR	Indexé	6*	2*
70 NEG	Étendu	7	3
71 *			
72 *			
73 COM		7	3
74 LSR		7	3
75 *			
76 ROR		7	3
77 ASR		7	3
78 ASL/LSL		7	3
79 ROL		7	3
7A DEC		7	3
7B *			
7C INC		7	3
7D TST		7	3
7E JMP		4	3
7F CLR	Étendu	7	3
80 SUBA	Immédiat	2	2
81 CMPA		2	2
82 SBCA		2	2
83 SUBD		4	3
84 ANDA		2	2
85 BITA		2	2
86 LDA		2	2
87 *			
88 EORA		2	2
89 ADCA		2	2
8A ORA		2	2
8B ADDA		2	2
8C CMPX	Immédiat	4	3
8D BSR	Relatif	7	2
8E LDX	Immédiat	3	3
8F *			

Légende

* nombre de cycles MPU
# nombre d'octets
* code OP non utilisé

Code OP	Mnémonique	Mode		#
90	SUBA	Direct	4	2
91	CMPA		4	2
92	SBCA		4	2
93	SUBD		6	2
94	ANDA		4	2
95	BITA		4	2
96	LDA		4	2
97	STA		4	2
98	EORA		4	2
99	ADCA		4	2
9A	ORA		4	2
9B	ADDA		4	2
9C	CMPX		6	2
9D	JSR		7	2
9E	LDX		5	2
9F	STX	Direct	5	2
A0	SUBA	Indexé	4+	2+
A1	CMPA		4+	2+
A2	SBCA		4+	2+
A3	SUBD		6+	2+
A4	ANDA		4+	2+
A5	BITA		4+	2+
A6	LDA		4+	2+
A7	STA		4+	2+
A8	EORA		4+	2+
A9	ADCA		4+	2+
AA	ORA		4+	2+
AB	ADDA		4+	2+
AC	CMPX		6+	2+
AD	JSR		7+	2+
AE	LDX		5+	2+
AF	STX	Indexé	5+	2+
B0	SUBA	Étendu	5	3
B1	CMPA		5	3
B2	SBCA		5	3
B3	SUBD		7	3
B4	ANDA		5	3
B5	BITA		5	3
B6	LDA		5	3
B7	STA		5	3
B8	EORA		5	3
B9	ADCA		5	3
BA	ORA		5	3
BB	ADDA		5	3
BC	CMPX		7	3
BD	JSR		8	3
BE	LDX		6	3
BF	STX	Étendu	6	3
C0	SUBB	Immédiat	2	2
C1	CMPB		2	2
C2	SBCB		2	2
C3	ADDD		4	3
C4	ANDB		2	2
C5	BITB	Immédiat	2	2

Code OP	Mnémonique	Mode		#
C6	LDB	Immédiat	2	2
C7	*			
C8	EORB		2	2
C9	ADCB		2	2
CA	ORB		2	2
CB	ADDB		2	2
CC	LDD		3	3
CD	*			
CE	LDU	Immédiat	3	3
CF	*			
D0	SUBB	Direct	4	2
D1	CMPB		4	2
D2	SBCB		4	2
D3	ADDD		6	2
D4	ANDB		4	2
D5	BITB		4	2
D6	LDB		4	2
D7	STB		4	2
D8	EORB		4	2
D9	ADCB		4	2
DA	ORB		4	2
DB	ADDB		4	2
DC	LDD		5	2
DD	STD		5	2
DE	LDU		5	2
DF	STU	Direct	5	2
E0	SUBB	Indexé	4+	2+
E1	CMPB		4+	2+
E2	SBCB		4+	2+
E3	ADDD		6+	2+
E4	ANDB		4+	2+
E5	BITB		4+	2+
E6	LDB		4+	2+
E7	STB		4+	2+
E8	EORB		4+	2+
E9	ADCB		4+	2+
EA	ORB		4+	2+
EB	ADDB		4+	2+
EC	LDD		5+	2+
ED	STD		5+	2+
EE	LDU		5+	2+
EF	STU	Indexé	5+	2+
F0	SUBB	Étendu	5	3
F1	CMPB		5	3
F2	SBCB		5	3
F3	ADDD		7	3
F4	ANDB		5	3
F5	BITB		5	3
F6	LDB		5	3
F7	STB		5	3
F8	EORB		5	3
F9	ADCB		5	3
FA	ORB		5	3
FB	ADDB	Étendu	5	3

Code OP	Mnémonique	Mode		#
FC	LDD	Étendu	6	3
FD	STD		6	3
FE	LDU		6	3
FF	STU	Indexé	6	3

Code OP	Mnémonique	Mode		#
1021	LBRN	Relatif	5	4
1022	LBHI		5(6)	4
1023	LBLS		5(6)	4
1024	LBHS / LBCC		5(6)	4
1025	LBCS / LBLO		5(6)	4
1026	LBNE		5(6)	4
1027	LBEQ		5(6)	4
1028	LBVC		5(6)	4
1029	LBVS		5(6)	4
102A	LBPL		5(6)	4
102B	LBMI		5(6)	4
102C	LBGE		5(6)	4
102D	LBLT		5(6)	4
102E	LBGT	Relatif	5(6)	4
102F	LBLE	Relatif	5(6)	4
103F	SWI 2	Implicite	20	2
1083	CMPD	Immédiat	5	4
108C	CMPY		5	4
108E	LDY	Immédiat	4	4
1093	CMPD	Direct	7	3
109C	CMPY		7	3
109E	LDY		6	3
109F	STY	Direct	6	3
10A3	CMPD	Indexé	7+	3+
10AC	CMPY		7+	3+
10AE	LDY		6+	3+
10AF	STY	Indexé	6+	3+
10B3	CMPD	Étendu	8	4
10BC	CMPY		8	4
10BE	LDY		7	4
10BF	STY	Étendu	7	4
10CE	LDS	Immédiat	4	4
10DE	LDS	Direct	6	3
10DF	STS	Direct	6	3
10EE	LDS	Indexé	6+	3
10EF	STS	Indexé	6+	3+
10FE	LDS	Étendu	7	4
10FF	STS	Étendu	7	4
113F	SWI 3	Implicite	20	2
1183	CMPU	Immédiat	5	4
118C	CMPS	Immédiat	5	4
1193	CMPU	Direct	7	3
119C	CMPS	Direct	7	3
11A3	CMPU	Indexé	7+	3+
11AC	CMPS	Indexé	7+	3+
11B3	CMPU	Étendu	8	4
11BC	CMPS	Étendu	8	4

NOTE : Tous les codes opération inutilisés sont indéfinis et interdits.

Tableau des instructions

Instruction	Forms	Addressing Modes: Immediate Op	Immediate ~	Immediate #	Direct Op	Direct ~	Direct #	Indexed Op	Indexed ~	Indexed #	Extended Op	Extended ~	Extended #	Inherent Op	Inherent ~	Inherent #	Description	5 H	3 N	2 Z	1 V	0 C
ABX														3A	3	1	B + X → X (Unsigned)	•	•	•	•	•
ADC	ADCA	89	2	2	99	4	2	A9	4+	2+	B9	5	3				A + M + C → A	↕	↕	↕	↕	↕
	ADCB	C9	2	2	D9	4	2	E9	4+	2+	F9	5	3				B + M + C → B	↕	↕	↕	↕	↕
ADD	ADDA	8B	2	2	9B	4	2	AB	4+	2+	BB	5	3				A + M → A	↕	↕	↕	↕	↕
	ADDB	CB	2	2	DB	4	2	EB	4+	2+	FB	5	3				B + M → B	↕	↕	↕	↕	↕
	ADDD	C3	4	3	D3	6	2	E3	6+	2+	F3	7	3				D + M M + 1 → D	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	4	2	A4	4+	2+	B4	5	3				A ∧ M → A	•	↕	↕	0	•
	ANDB	C4	2	2	D4	4	2	E4	4+	2+	F4	5	3				B ∧ M → B	•	↕	↕	0	•
	ANDCC	1C	3	2													CC ∧ IMM → CC	?	?	?	?	?
ASL	ASLA													48	2	1		↕	↕	↕	↕	↕
	ASLB													58	2	1		↕	↕	↕	↕	↕
	ASL				08	6	2	68	6+	2+	78	7	3					↕	↕	↕	↕	↕
ASR	ASRB													47	2	1		↕	↕	↕	•	↕
	ASR													57	2	1		↕	↕	↕	•	↕
	ASR				07	6	2	67	6+	2+	77	7	3					↕	↕	↕	•	↕
BIT	BITA	85	2	2	95	4	2	A5	4+	2+	B5	5	3				Bit Test A (M ∧ A)	•	↕	↕	0	•
	BITB	C5	2	2	D5	4	2	E5	4+	2+	F5	5	3				Bit Test B (M ∧ B)	•	↕	↕	0	•
CLR	CLRA													4F	2	1	0 → A	•	0	1	0	0
	CLRB													5F	2	1	0 → B	•	0	1	0	0
	CLR				0F	6	2	6F	6+	2+	7F	7	3				0 → M	•	0	1	0	0
CMP	CMPA	81	2	2	91	4	2	A1	4+	2+	B1	5	3				Compare M from A	↕	↕	↕	↕	↕
	CMPB	C1	2	2	D1	4	2	E1	4+	2+	F1	5	3				Compare M from B	↕	↕	↕	↕	↕
	CMPD	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M M + 1 from D	•	↕	↕	↕	↕
		83			93			A3			B3											
	CMPS	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M M + 1 from S	•	↕	↕	↕	↕
		8C			9C			AC			BC											
	CMPU	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M M + 1 from U	•	↕	↕	↕	↕
		83			93			A3			B3											
	CMPX	8C	4	3	9C	6	2	AC	6+	2+	BC	7	3				Compare M M + 1 from X	•	↕	↕	↕	↕
	CMPY	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M M + 1 from Y	•	↕	↕	↕	↕
		8C			9C			AC			BC											
COM	COMA													43	2	1	$\overline{A} \rightarrow A$	•	↕	↕	0	1
	COMB													53	2	1	$\overline{B} \rightarrow B$	•	↕	↕	0	1
	COM				03	6	2	63	6+	2+	73	7	3				$\overline{M} \rightarrow M$	•	↕	↕	0	1
CWAI		3C	≥20	2													CC ∧ IMM → CC Wait for Interrupt					?
DAA														19	2	1	Decimal Adjust A	•	↕	↕	0	↕
DEC	DECA													4A	2	1	A − 1 → A	•	↕	↕	↕	•
	DECB													5A	2	1	B − 1 → B	•	↕	↕	↕	•
	DEC				0A	6	2	6A	6+	2+	7A	7	3				M − 1 → M	•	↕	↕	↕	•
EOR	EORA	88	2	2	98	4	2	A8	4+	2+	B8	5	3				A ⊻ M → A	•	↕	↕	0	•
	EORB	C8	2	2	D8	4	2	E8	4+	2+	F8	5	3				B ⊻ M → B	•	↕	↕	0	•
EXG	R1, R2	1E	8	2													R1 ↔ R2[2]	•	•	•	•	•
INC	INCA													4C	2	1	A + 1 → A	•	↕	↕	↕	•
	INCB													5C	2	1	B + 1 → B	•	↕	↕	↕	•
	INC				0C	6	2	6C	6+	2+	7C	7	3				M + 1 → M	•	↕	↕	↕	•
JMP					0E	3	2	6E	3+	2+	7E	4	3				EA[3] → PC	•	•	•	•	•
JSR					9D	7	2	AD	7+	2+	BD	8	3				Jump to Subroutine	•	•	•	•	•
LD	LDA	86	2	2	96	4	2	A6	4+	2+	B6	5	3				M → A	•	↕	↕	0	•
	LDB	C6	2	2	D6	4	2	E6	4+	2+	F6	5	3				M → B	•	↕	↕	0	•
	LDD	CC	3	3	DC	5	2	EC	5+	2+	FC	6	3				M M + 1 → D	•	↕	↕	0	•
	LDS	10	4	4	10	6	3	10	6+	3+	10	7	4				M M + 1 → S	•	↕	↕	0	•
		CE			DE			EE			FE											
	LDU	CE	3	3	DE	5	2	EE	5+	2+	FE	6	3				M M + 1 → U	•	↕	↕	0	•
	LDX	8E	3	3	9E	5	2	AE	5+	2+	BE	6	3				M M + 1 → X	•	↕	↕	0	•
	LDY	10	4	4	10	6	3	10	6+	3+	10	7	4				M M + 1 → Y	•	↕	↕	0	•
		8E			9E			AE			BE											
LEA	LEAS							32	4+	2+							EA[3] → S	•	•	•	•	•
	LEAU							33	4+	2+							EA[3] → U	•	•	•	•	•
	LEAX							30	4+	2+							EA[3] → X	•	•	↕	•	•
	LEAY							31	4+	2+							EA[3] → Y	•	•	↕	•	•

OP	Code opération (hexadécimal)	Z	Zéro (RESET)
-	Nombre de cycles MPU	V	Débordement, complément de 2
#	Nombre d'octets programme	C	Retenue de l'UAL
+	Addition arithmétique	|	Test/fixé si vrai, sinon RAZ
−	Soustraction arithmétique	.	Non affecté
×	Multiplication	CC	Registre code état
$\overline{M}$	Complément de M	:	Concaténation
→	Transfert vers	∨	OU logique
H	Demi-retenue (de bit 3)	∧	ET logique
N	Négatif (bit de signe)	⊻	OU exclusif logique

Tableau des instructions (suite)

Instruction	Forms	Addressing Modes: Immediate Op	~	#	Direct Op	~	#	Indexed[1] Op	~	#	Extended Op	~	#	Inherent Op	~	#	Description	5 H	3 N	2 Z	1 V	0 C
LSL	LSLA													48	2	1	A B M ← 0 ← b7 b0 C	•	↕	↕	↕	↕
	LSLB													58	2	1		•	↕	↕	↕	↕
	LSL				08	6	2	68	6+	2+	78	7	3					•	↕	↕	↕	↕
LSR	LSRA													44	2	1	A B M 0 → b7 b0 → C	•	0	↕	•	↕
	LSRB													54	2	1		•	0	↕	•	↕
	LSR				04	6	2	64	6+	2+	74		3					•	0	↕	•	↕
MUL														3D	11	1	A × B → D (Unsigned)	•	•	↕	•	9
NEG	NEGA													40	2	1	$\overline{A}$ + 1 → A	⇕	↕	↕	↕	↕
	NEGB													50	2	1	$\overline{B}$ + 1 → B	⇕	↕	↕	↕	↕
	NEG				00	6	2	60	6+	2+	70	7	3				$\overline{M}$ + 1 → M	⇕	↕	↕	↕	↕
NOP														12	2	1	No Operation	•	•	•	•	•
OR	ORA	8A	2	2	9A	4	2	AA	4+	2+	BA	5	3				A ∨ M → A	•	↕	↕	0	•
	ORB	CA	2	2	DA	4	2	EA	+	2+	FA	5	3				B ∨ M → B	•	↕	↕	0	•
	ORCC	1A	3	2													CC ∨ IMM → CC	↕	↕	↕	↕	↕
PSH	PSHS	34	5+[4]	2													Push Registers on S Stack	•	•	•	•	•
	PSHU	36	5+[4]	2													Push Registers on U Stack	•	•	•	•	•
PUL	PULS	35	5+[4]	2													Pull Registers from S Stack	•	•	•	•	•
	PULU	37	5+[4]	2													Pull Registers from U Stack	•	•	•	•	•
ROL	ROLA													49	2	1	A B M ← C ← b7 b0	•	↕	↕	↕	↕
	ROLB													59	2	1		•	↕	↕	↕	↕
	ROL				08	6	2	69	6+	2+	79	7	3					•	↕	↕	↕	↕
ROR	RORA													46	2	1	A B M → C → b7 b0	•	↕	↕	•	↕
	RORB													56	2	1		•	↕	↕	•	↕
	ROR				06	6	2	66	6+	2+	76	7	3					•	↕	↕	•	↕
RTI														3B	6/15	1	Return From Interrupt					7
RTS														39	5	1	Return from Subroutine	•	•	•	•	•
SBC	SBCA	82	2	2	92	4	2	A2	4+	2+	B2	5	3				A − M − C → A	⇕	↕	↕	↕	↕
	SBCB	C2	2	2	D2	4	2	E2	4+	2+	F2	5	3				B − M − C → B	⇕	↕	↕	↕	↕
SEX														1D	2	1	Sign Extend B into A	•	↕	↕	0	•
ST	STA				97	4	2	A7	4+	2+	B7	5	3				A → M	•	↕	↕	0	•
	STB				D7	4	2	E7	4+	2+	F7	5	3				B → M	•	↕	↕	0	•
	STD				DD	5	2	ED	5+	2+	FD	6	3				D → M M+1	•	↕	↕	0	•
	STS				10	6	3	10	6+	3+	10	7	4				S → M M+1	•	↕	↕	0	•
					DF			EF			FF											
	STU				DF	5	2	EF	5+	2+	FF	6	3				U → M M+1	•	↕	↕	0	•
	STX				9F	5	2	AF	5+	2+	BF	6	3				X → M M+1	•	↕	↕	0	•
	STY				10	6	3	10			10	7	4				Y → M M+1	•	↕	↕	0	•
					9F			AF	6+	3+	BF											
SUB	SUBA	80	2	2	90	4	2	A0	4+	2+	B0	5	3				A − M → A	⇕	↕	↕	↕	↕
	SUBB	C0	2	2	D0	4	2	E0	4+	2+	F0	5	3				B − M → B	⇕	↕	↕	↕	↕
	SUBD	83	4	3	93	6	2	A3	6+	2+	B3	7	3				D − M M+1 → D	•	↕	↕	↕	↕
SWI	SWI[6]													3F	19	1	Software Interrupt 1	•	•	•	•	•
	SWI2[6]													10 3F	20	2	Software Interrupt 2	•	•	•	•	•
	SWI3[6]													11 3F	20	1	Software Interrupt 3	•	•	•	•	•
SYNC														13	≥4	1	Synchronize to Interrupt	•	•	•	•	•
TFR	R1, R2	1F	8	2													R1 → R2[2]	•	•	•	•	•
TST	TSTA													4D	2	1	Test A	•	↕	↕	0	•
	TSTB													5D	2	1	Test B	•	↕	↕	0	•
	TST				0D	6	2	6D	6+	2+	7D	7	3				Test M	•	↕	↕	0	•

Notes

1. Cette colonne donne le nombre de cycles de base et le décompte d'octets. Afin d'obtenir le décompte total, ajouter les valeurs obtenues dans la table Mode d'adressage indexé.
2. R1 et R2 doivent être une paire de registres 8 bits ou une paire de registres 16 bits.
3. EA est l'adresse effective.
4. Les instructions PSH et PUL nécessitent cinq cycles plus un cycle pour chaque octet empilé ou dépilé.
5. 5 (6) signifie : cinq cycles si branchement non pris, six cycles si branchement pris (instructions de branchement).
6. SWI1 fixe les bits F et I. SWI1 et SWI2 n'affectent pas I et F.
7. L'état du registre code condition résulte directement de cette instruction.
8. Valeur du flag de demi-retenue non défini.
9. Cas particulier : retenue de bit fixé si B7 est SET.

Instruction	Forms	Addressing Mode: Relative OP	~	#	Description	5 H	3 N	2 Z	1 V	0 C
BCC	BCC	24	3	2	Branch C = 0	•	•	•	•	•
	LBCC	10	5(6)	4	Long Branch	•	•	•	•	•
		24			C = 0					
BCS	BCS	25	3	2	Branch C = 1	•	•	•	•	•
	LBCS	10	5(6)	4	Long Branch	•	•	•	•	•
		25			C = 1					
BEQ	BEQ	27	3	2	Branch Z = 1	•	•	•	•	•
	LBEQ	10	5(6)	4	Long Branch	•	•	•	•	•
		27			Z = 1					
BGE	BGE	2C	3	2	Branch ≥ Zero	•	•	•	•	•
	LBGE	10	5(6)	4	Long Branch ≥ Zero	•	•	•	•	•
		2C								
BGT	BGT	2E	3	2	Branch > Zero	•	•	•	•	•
	LBGT	10	5(6)	4	Long Branch > Zero	•	•	•	•	•
		2E								
BHI	BHI	22	3	2	Branch Higher	•	•	•	•	•
	LBHI	10	5(6)	4	Long Branch Higher	•	•	•	•	•
		22								
BHS	BHS	24	3	2	Branch Higher or Same	•	•	•	•	•
	LBHS	10	5(6)	4	Long Branch Higher or Same	•	•	•	•	•
		24								
BLE	BLE	2F	3	2	Branch ≤ Zero	•	•	•	•	•
	LBLE	10	5(6)	4	Long Branch ≤ Zero	•	•	•	•	•
		2F								
BLO	BLO	25	3	2	Branch lower	•	•	•	•	•
	LBLO	10	5(6)	4	Long Branch Lower	•	•	•	•	•
		25								
BLS	BLS	23	3	2	Branch Lower or Same	•	•	•	•	•
	LBLS	10	5(6)	4	Long Branch Lower or Same	•	•	•	•	•
		23								
BLT	BLT	2D	3	2	Branch < Zero	•	•	•	•	•
	LBLT	10	5(6)	4	Long Branch < Zero	•	•	•	•	•
		2D								
BMI	BMI	2B	3	2	Branch Minus	•	•	•	•	•
	LBMI	10	5(6)	4	Long Branch Minus	•	•	•	•	•
		2B								
BNE	BNE	26	3	2	Branch Z = 0	•	•	•	•	•
	LBNE	10	5(6)	4	Long Branch	•	•	•	•	•
		26			Z = 0					
BPL	BPL	2A	3	2	Branch Plus	•	•	•	•	•
	LBPL	10	5(6)	4	Long Branch Plus	•	•	•	•	•
		2A								
BRA	BRA	20	3	2	Branch Always	•	•	•	•	•
	LBRA	16	5	3	Long Branch Always	•	•	•	•	•
BRN	BRN	21	3	2	Branch Never	•	•	•	•	•
	LBRN	10	5	4	Long Branch Never	•	•	•	•	•
		21								
BSR	BSR	8D	7	2	Branch to Subroutine	•	•	•	•	•
	LBSR	17	9	3	Long Branch to Subroutine	•	•	•	•	•
BVC	BVC	28	3	2	Branch V = 0	•	•	•	•	•
	LBVC	10	5(6)	4	Long Branch	•	•	•	•	•
		28			V = 0					
BVS	BVS	29	3	2	Branch V = 1	•	•	•	•	•
	LBVS	10	5(6)	4	Long Branch	•	•	•	•	•
		29			V = 1					

SIMPLE BRANCHES

	OP	~	#
BRA	20	3	2
LBRA	16	5	3
BRN	21	3	2
LBRN	1021	5	4
BSR	8D	7	2
LBSR	17	9	3

SIMPLE CONDITIONAL BRANCHES (Notes 1-4)

Test	True	OP	False	OP
N = 1	BMI	2B	BPL	2A
Z = 1	BEQ	27	BNE	26
V = 1	BVS	29	BVC	28
C = 1	BCS	25	BCC	24

SIGNED CONDITIONAL BRANCHES (Notes 1-4)

Test	True	OP	False	OP
r > m	BGT	2E	BLE	2F
r ≥ m	BGE	2C	BLT	2D
r = m	BEQ	27	BNE	26
r ≤ m	BLE	2F	BGT	2E
r < m	BLT	2D	BGE	2C

UNSIGNED CONDITIONAL BRANCHES (Notes 1-4)

Test	True	DP	False	OP
r > m	BHI	22	BLS	23
r ≥ m	BHS	24	BLO	25
r = m	BEQ	27	BNE	26
r ≤ m	BLS	23	BHI	22
r < m	BLO	25	BHS	24

Notes

1. Tous les branchements conditionnels ont des variantes longues et courtes.
2. Tous les branchements courts sont de deux octets et nécessitent trois cycles.
3. Tous les branchements conditionnels longs sont constitués en préfixant $ 1Ø au code opération branchement court et utilisent un déplacement de destination de seize bits.
4. Tous les branchements conditionnels longs nécessitent quatre octets et six cycles si le branchement est pris, ou cinq cycles si le branchement n'est pas pris.

ANNEXE C
CODE ASCII

b7 b6 b5 / Bits						0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b4	b3	b2	b1	Column		0	1	2	3	4	5	6	7
1	1	1	1	Row	Hex	0	1	2	3	4	5	6	7
0	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	1	SDH	DC1	!	1	A	Q	a	q
0	0	1	0	2	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	4	EQT	DC4	$	4	D	T	d	t
0	1	0	1	5	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	C	FF	FS	,	<	L	\	l	|
1	1	0	1	13	D	CR	GS	-	=	M	)	m	}
1	1	1	0	14	E	SD	RS	.	>	N	^	n	~
1	1	1	1	15	F	SI	US	/	?	D	_	o	DEL

Caractères de contrôle

NUL	Null
SOH	Start of Heading
STX	Start of Text
ETX	End of Text
EOT	End of Transmission
ENQ	Enquiry
ACK	Acknowledge
BEL	Bell
BS	Backspace
HT	Horizontal Tabulation
LF	Line Feed
VT	Vertical Tabulation
FF	Form Feed
CR	Carriage Return
SO	Shift Out

SI	Shift In
DLE	Data Link Escape
DC1	Device Control 1
DC2	Device Control 2
DC3	Device Control 3
DC4	Device Control 4
NAK	Negative Acknowledge
SYN	Synchronous Idle
ETB	End of Transmission Block
CAN	Cancel
EM	End of Medium
SUB	Substitute
ESC	Escape
FS	File Separator
GS	Group Separator
RS	Record Separator
US	Unit Separator
DEL	Delete

ANNEXE D
MESSAGES D'ERREUR

Les erreurs d'entrée/sortie

- **« Bad Parameters »**
Le paramètre de la commande est incorrect.
(Erreur dans le numéro du disque).

- **« Disk not ready »**
Le disque n'est pas prêt.

- **« Bad file descriptor »**
Erreur de syntaxe dans le descripteur du fichier ou descripteur absent.

- **« I/O Error »**
Impossibilité physique de réaliser une entrée/sortie.

- **« File Format Error »**
Le fichier ne peut pas être chargé. Il y a deux causes possibles :
- erreur sur la disquette,
- le mode de chargement utilisé est incorrect.

- **« File not Found »**
Le fichier décrit par le descripteur n'est pas trouvé sur le périphérique indiqué.

- **« Verification Error »**
Le contenu de la mémoire est différent du contenu du fichier indiqué.

- **« Bad Memory »**
L'adresse indiquée pour l'implantation du code objet est incorrecte.

- **« File Already Exists »**
Le nom du fichier ne peut pas être remplacé par un nom déjà existant dans le catalogue (RENAME).

- **« File Already Open »**
Le fichier ne peut pas être copié sur lui-même.

- **« Disk Write Protected »**
Essai d'écriture sur une disquette protégée.

- **« Out Of Memory »**
Il n'y a pas suffisamment de RAM libre.

- « **Disk Full** »
Il n'y a plus de place sur la disquette.

- « **Unreadable Disk** »
La disquette utilisée n'a pas été formatée.

- « **Directory Full** »
Il n'y a plus de place dans le catalogue de la disquette.

Les erreurs d'assemblage

- « **Branch Out Of Range** »
L'adresse du branchement est trop éloignée de l'adresse de l'instruction courante.

- « **Bad Address** »
L'adresse indiquée pour l'implantation du code objet lors de l'assemblage en mémoire est incorrecte.

- « **Bad Operand** »
L'opérande n'est pas reconnu par l'assembleur.

- « **Bad Includ** »
Le fichier indiqué par INCLUD se trouve dans la cassette ou contient lui-même une autre directive INCLUD.

- « **Expression Error** »
Mauvaise expression utilisée dans le champ opérande.

- « **Bad Opcode** »
Mauvais code opération.

- « **Undefined symbol** »
Le symbole utilisé n'a pas été défini.

- « **Missing END Statement** »
Le programme source n'est pas terminé par la directive END.

- « **Multiply defined symbol** »
Le même symbole est utilisé plusieurs fois.

- « **Register Error** »
Le symbole utilisé ne correspond pas à un registre du microprocesseur.

- « **Bad Label** »
Mauvaise étiquette.

- « **Opérand Too Large** »
Operande trop grand.

- « **DP Error** »
Erreur sur la page directe.

- « **Symbol Table Full** »
Il n'y a plus de place dans la table des symboles.

Les autres erreurs possibles sous éditeur-moniteur ou Entrées/Sorties

- « **Bad Command** »
La commande n'est pas reconnue par le système.

- « **Bad Parameter(s)** »
Le paramètre de la commande est incorrect ou absent.

- « **Missing operand** »
Il manque au moins un paramètre dans les commandes **SAVE** et **VERIFY** sous contrôle du moniteur.

- « **Undefined Symbol** »
Le symbole utilisé n'est pas reconnu par le moniteur.

- « **String not found** »
La chaîne de caractères n'a pas été trouvée.

- « **Internal Error** »
Erreur interne *

- « **Missing information** »
Il manque une information dans la commande.

- « **Can't Continue** »

Le programme en cours est arrêté et ne peut plus être repris par **C (CONTINUE)** car il est en ROM.

- « **Bad Breakpoint** »
Erreur dans les commandes des points d'arrêt.

*N.B. : Cette erreur est généralement due à des effets incontrôlés du programme utilisateur. Une fois le programme sauvé, il peut être prudent de réinitialiser tout le système en coupant l'alimentation car des registres propres à l'Assembleur peuvent avoir été altérés.

ANNEXE E

PROGRAMMES TYPES

Programme type A

1

```
*****************************************
* Programme de balayage aleatoire de    *
*          la memoire Point             *
*****************************************
* Une frappe sur une touche quelconque *
* modifie le dessin .                   *
*****************************************
* Pour sortir du Programme tapez ENTREE*
*****************************************

        ORG     ENDMEM-$400 1K Reserve
DIRECT  EQU     *<-8    Page 0 ici
        SETDP   DIRECT
*
        TITLE   Balayage ecran.
*
        INCLUD EQUATE  Fichier contenant
* les principaux points d'entree du
* Moniteur MO5
```

Note : en version cassette, INCLUD doit être remplacé par le contenu du fichier EQUATES.

```
CNST    FDB     1       Constante dep-
* lacement,initialisee a 1, ou les car-
* acteres entres au clavier vont rentrer
* par la droite.

ACC     RMB     2       Accumulateur de
* calcul

TABIT   FCB     $80,$40,20,$10,$08,$04
        FCB     $02,$01
* Table des bits qui vont etre inverses
* en memoire.C'est un bit a 1 qui occu-
* pe successivement les 8 positions d'un
* octet.
```

2

```
COUL0   SET     $41
COUL1   SET     $5F
COUL2   SET     $64
COUL    FCB     ESC,COUL0,ESC
        FCB     COUL1,ESC
        FCB     COUL2,FF,EOT
* Sequence de codes pour mettre l'ecran
* en ROUGE sur fond ORANGE,avec tour
* BLEU et effacement (FORM FEED). EOT
* indique la fin de la sequence .

LIGB    FCB     US,12H,14H,EOT
LIGH    FCB     US,20H,20H,EOT
* Sequence de codes pour elargir la
* fenetre a l'ensemble de l'ecran.

        PAGE
START   PSHS    A,B,X,Y,U,DP Sauvegarde
        LDA     #DIRECT Page 0
        TFR     A,DP
        LDX     #LIGB   Elargit la fenetre
        JSR     DISPL
        LDX     #LIGH
        JSR     DISPL
NEW     LDX     #COUL   Codes qui mettent
* l'ecran en COUL et effacent tout.

        JSR     DISPL   Routine d'afficha-
* ge d'une sequence de codes .
```

3

```
        LDA     PORTA   MISE en memoire
* Points Par mise a 1 du bit 0 du
* Port A(EQUATE)

        ORA     #1

        STA     PORTA
        LDX     #STADM  Adresse de debut
* de l'ecran (EQUATE)

        CLR     ACC+1
LOOP    CLRA            Boucle generale de
* Parcourt d'un ecran complet,Comme A
* vaut 0 D=[A,B] vaut de 0 a 256 .

        LDB     ACC+1   L'ensemble de 16
* bits [ACC,ACC+1] va servir de registre
* de calcul de l'adresse dans l'ecran .
* [ACC+1] donne la coordonnee X, et le
* OU exclusif entre ses deux octets don-
* ne la coordonnee Y.C'est l'algorithme
* qui sert a balayer l'ecran de maniere
* Pseudo-aleatoire .

        ADDD    #32     Pour faire une
* image centree Y (ligne) va de 0 a
* 199 et X (colonne) va de 32 a 287,
* c'est un rectangle de 200 * 256 .

        TFR     D,X     X=colonne
        LDB     ACC     OU exclusif
        EORB    ACC+1
        CMPD    #200    Si le resultat est
* superieur a 199 (ligne maximum), on
* ignore ce tour et on continue la bou-
* cle

        BHS     SAUTE
        TFR     D,Y     Y=ligne
        BSR     CALCUL  Routine qui trans-
* forme les coordonnees logiques X et Y
* en une adresse Physique dans la memoi-
* re ecran du MO5.Cette adresse est ret-
* ournee Par le registre X qui pointe
* sur l'octet recherche.Le point dans
* l'octet est donne Par les 3 derniers
* bits du no de colonne: les 3 LSB de
* ACC+1 transformes en bit par TABIT.
        LDY     #TABIT  Table de transfor-
* mation de [0,7] en un bit a 1 .
        LDA     ACC+1   On transforme le
* 3 bits de droite de l'adresse colonne
```

```
        ANDA    #7
        LDA     A,Y       Offset Par 3 LSB
        EORA    ,X        Seul le but a 1
* dans A inversera le bit correspondant
* de l'adresse [X,Y].Le OU exclusif
* avec les autres bits a 0 est sans ef-
* fet .

        STA     ,X
SAUTE   LDD     ALL       D=[ALL,ALL+1]
        ADDD    CNST      La constante CNST
* fait passer a l'octet suivant elle
* s'ajoute a X et definit ainsi l'incre-
* ment du Parcours de l'ecran.Il y a
* 65536 Possibilites.

        STD     ALL
        LEAU    4,U       Frequence de lect-
* ure du clavier: tous les 65536/4 tours
* Le clavier est lu et si une touche est
* enfoncee elle modifie la constante
* CNST.donc le Pas de balayage et le
* resultat sur l'ecran .

        CMPU    #4
        BHS     LOOP
        CALL    KTST      Test rapide d'une
* eventuelle touche enfoncee (EQUATE).

        BCC     LOOP      Si non, continue.
        CALL    GETCH     si oui, lire la
* touche enfoncee (EQUATE).

        TSTB
        BEQ     LOOP      Elle a ete rela-
* chee entre temps .

        CMPB    #CR       Touche ENTREE ?
        BEQ     OUT       Oui, on sort.
L2      ANDB    #$F       Sinon,on ne garde
* que les 4 bits de droite du code de la
* touche, on decale CNTS (16bits) de 4
* bits vers la gauche et on rentre les 4
* bits nouveaux Par la droite.

        PSHS    B         Sauver les 4 bits
        LDD     CNST
        ASLB              Decaler 4 fois
        ROLB
        ASLB
        ROLB
        ASLB
        ROLB
        ASLB
        ROLB
        ADDB    ,S+       Ajouter les 4 bits
        STD     CNST
        JMP     NEW       On recommence un
* nouvel ecran avec la nouvelle valeur
* de CNST.

OUT     PULS    A,B,X,,U,PC Sortie
        STOP              Retour au moniteur
        PAGE

*
* Sous Programme de conversion d'une ad-
* resse logique dans [X,Y] en une adres-
* se physique retournee Par X .

CALCUL PSHS     A,B,Y
        TFR     Y,D       D=ligne
        LDA     #40
        MUL
        ADDD    #STADR
        EXG     D,X       X=ligne * 40 +
* Adresse de debut de l'ecran, D=colonne
        LSRA
        RORB
        LSRB
        LSRB              D=colonne/8
        LEAX    D,X       + Ligne * 40
        PULS    A,B,Y,PC Retour au Pro-
* gramme appelant avec l'adresse dans X.
*
* Sous Programme affichant une chaine de
* codes Pointee Par X et terminee Par le
* code EOT (defini dans EQUATE).

CONT    CALL    PUTCH     Point d'entree de
* la routine d'affichage du moniteur MO5
* defini dans EQUATE
DISPL   LDB     ,X+       B=code Pointe
* Par X
        CMPB    #EOT      Fin de chaine
        BNE     CONT      Sinon, affichage
        RTS

        END     START
```

Programme type B

Le programme "balayage aléatoire de la mémoire couleur" peut être obtenu à partir du programme ci-dessus en remplaçant les parties 1 2 3 par 1 2 3 ci-dessous.

1

```
****************************************
* Programme de balayage aleatoire de   *
*          la memoire couleur          *
****************************************
* Une frappe sur une touche quelconque *
* modifie le dessin .                  *
****************************************
* Pour sortir du programme tapez ENTREE*
****************************************
```

2

```
NOIR    FCB     ESC,$43,ESC,$55,ESC,$62
        FCB     FF,EOT
* Sequence de codes Pour mettre l'ecran
* en jaune sur fond violet,avec tour
* vert et effacement (FORM FEED). EOT
* indique la fin de la sequence .
```

3

```
        LDA     PORTA     MISE en memoire
* Couleur Par mise a 0 du bit 0 du
* Port C(EQUATE)
        ANDA    #$FE
```

Programme type C

```
********** OLYMPIQUE ******************
*****************************************
* APPel le sous Programme cercle        *
*****************************************
* Note de l'auteur:Ce programme n'a     *
* d'utilite que de mettre en valeur le  *
* sous Programme cercle .               *
*****************************************
        ORG     ENDMEM-&1024
FORME   EQU     $2029
INICHA  FCB     $1B,$66,$1B,$5D,$1B,$4F
        FCB     $0C,$14,$0D
TEXCHA  FCB     $1F,$41,$46,$1B,$73H
        FCC     /LOS ANGELES 84/
        FCB     $1F,$40+&24,$40+&12
        FCC     /SEOUL 88/
        FCB     $1E,$1B,$70,$0D
EFFA    FCB     0CH,0DH
LIGB    FCB     1FH,12H,14H,0DH
LIGH    FCB     1FH,20H,20H,0DH
DEBUT   LDX     #LIGB
        JSR     AFCHAI
        LDX     #LIGH
        JSR     AFCHAI
        LDX     #EFFA
        JSR     AFCHAI
        LDU     #ENDMEM-1
        LDX     #INICHA
        JSR     AFCHAI
        LDB     #&37    Parametres d'entree
* du s/P cercle
        LDY     #&79
        LDX     #&68
        LDA     #&4
        BSR     AFCER
        LDA     #&0
        BSR     AFCER
        LDA     #&1
        BSR     AFCER
        LDX     #&115
        LDY     #&119
        LDA     #&3
        BSR     AFCER
        LDA     #&2
        BSR     AFCER
        LDX     #TEXCHA
        JSR     AFCHAI
        CALL    $0C
        BCC     *-2
        STOP

AFCER   PSHU    B
        STA     FORME
KL      BSR     CERCLE
        INCB
        CMPB    #&42
        BLS     KL
        LEAX    &88,X

        PULU    B
        RTS



*CERCLE*********************************
*         -Rayon dans B                 *
*         -Coordonnee :  x dans X       *
*                        y dans Y       *
*****************************************

* initialisation **********************
CHRAW   EQU     $2036
PLOT    EQU     $10H
RCAR    RMB     2       rayon au carre
MX      RMB     1       abscisse
MYC     RMB     2       abscisse au carre
MY      RMB     1       ordonnee
CERCLE  PSHU    Y,X,DP,B,A,CC
        SETDP   #<-8
        LDA     ##<-8
        TFR     A,DP
        TSTB
        BMI     FG4
        CLR     MY
        CLR     CHRAW

* calcul du rayon au carre ************
        TFR     B,A
        MUL
        STD     RCAR    16 bits

* calcul du rayon multiPlie Par 3/4 ****
        LDB     2,U
        LSRB
        STB     MX
        LSRB
        ADDB    MX
        STB     MX      8 bits

* calcul de MX au carre Puis compLemen-
* tation a 2
FG0     LDB     MX
        TFR     B,A
        MUL
        COMA
        COMB
        ADDD    #$0001

* calcul de Y carre et stockage dans MYC
        ADDD    RCAR
        STD     MYC     16 bits

* calcul de Y a Partir de Y carre
FG1     LDB     MY
        TFR     B,A
        MUL
        CMPD    MYC
        BHS     FG2
        INC     MY
        BRA     FG1

* affichage des Points ****************

FG2     BSR     AFFI
        LDA     MX
        LDB     MY
        STA     MY
        STB     MX
        BSR     AFFI
        LDB     MX
        LDA     MY
        STB     MY
        DECA
        STA     MX
        BPL     FG0
FG4     PULU    Y,X,DP,B,A,CC
        RTS

AFFI    BSR     AFFI1
        NEG     MX
        BSR     AFFI1
        NEG     MY
        BMI     AFFI
        RTS

AFFI1   LDB     MX
        SEX
        ADDD    4,U
        CMPD    #&319
        BHI     FG3
        TFR     D,X
        LDB     MY
        SEX
```

```
        ADDD    6,U
        CMPD    #&199
        BHI     FG3
        TFR     D,Y
        CALL    PLOT
FG3     RTS


**AFFICHAGE D'UNE CHAINE DE CARACTERES**
*adresse du Premier caractere dans X   *
*La chaine doit se terminer par CR (0D)*
*si C=1 il y a retour du curseur a la  *
* ligne apres l'affichage               *
*Tous les registres sont sauvegardes   *
****************************************
PUTC    EQU     $02       sous-Prog.af.carr.
AFCHAI  PSHU    B,X       sauvegarde de B,X
        PSHU    CC        sauvegarde le RCC
DR1     LDB     ,X+       charge B et X=X+1
        CMPB    #$0D      teste la fin de
* chaine
        BEQ     DR0       si oui --> DR0
        CALL    PUTC      si non > affichage
        BRA     DR1       retour a boucle
DR0     PULU    CC        depile C Pour test
        BCC     DR2       si C=0->DR2
        CALL    PUTC      si C=1 retour du
* curseur a la ligne
        LDB     #$0A      descente d'une
* ligne
        CALL    PUTC
DR2     PULU    B,X       depile B et X
        RTS               fin
****************************************
```

Programme type D

```
****************************************
* BRIAN                                *
* ------------------------------------ *
* Pour arreter Brian tapez ENTREE      *
* ------------------------------------ *
* Une nouvelle frappe ENTREE permet de*
* sortir de Brian                      *
****************************************
        ORG     ENDMEM-$4000
DIRECT  EQU     *<-8
        SETDP   DIRECT
CNST    FDB     1         Constante
ACC     RMB     2         Accumulateur
TABIT   FDB     $7FBF,$DFDE,$F7FB,$FDFE
NOIR    FCB     0CH,04T
LIGB    FCB     1FH,12H,14H,04H
LIGH    FCB     1FH,20H,20H,04H
CONT    CALL    $02       Point d'entree de
DISPL   LDB     ,X+       B=code Pointe
        CMPB    #04H      Fin de chaine
        BNE     CONT      Sinon, affichage
        RTS
START   PSHS    A,B,X,Y,U,DP
        LDA     #DIRECT
        TFR     A,DP
        LDX     #LIGB
        JSR     DISPL
        LDX     #LIGH
        JSR     DISPL
        LDX     #NOIR
        JSR     DISPL
NEW     CLR     $A7C0     COLOUR
        LDB     #$0C
        CALL    $02       PUTCH
        CALL    $06       Mise en forme
        LDX     #$0000    STAD
M2      LDY     #20       Compteur ligne
        LDD     #$55AA    Moirage
M0      STD     ,X++
        LEAY    -1,Y
        BNE     M0
        LDY     #20       Ligne suivante
        LDD     #$AA55
M1      STD     ,X++
        LEAY    -1,Y
        BNE     M1
        CMPX    #$1F40    Fin d'ecran ?
        BLS     M2
        DEC     $A7C0     Couleur
        CLR     ACC       Accumulateur
        CLR     ACC+1
LOOP    CLRA
        LDB     ACC+1
        ADDD    #32       Pour centrer
        TFR     D,X       X=colonne (ACCL)
        LDB     ACC
        EORB    ACC+1
        CMPB    #200      Fin de l'ecran ?
        BHS     SKIP
        TFR     D,Y       Y=ligne
* Y = ACCH XOR ACCL
        BSR     CALCUL
        LDY     #TABIT
        LDA     ACC+1
        ANDA    #7
        LDA     A,Y       Offset Par 3 LSB
        EORA    ,X        Inverse les bits
* de l'adresse [X,Y]
        STA     ,X
SKIP    LDD     ACC       La constante fait
* Passer a l'octet suivant
        ADDD    CNST
        STD     ACC
        LEAU    32T,U     Frequence de lect-
        CMPU    #1        ure du clavier
        BHS     LOOP
        CALL    $0C       KTST
        BCC     LOOP
        CALL    $0A       GETCH
        BEQ     LOOP
        CMPB    #$D       CR ?
        BEQ     OUT
        CMPB    #$40
        BCS     L2
        ADDB    #9
L2      ANDB    #$F
        PSHS    B
        LDD     CNST
        ASLB
        ROLA
        ASLB
        ROLA
        ASLB
        ROLA
        ASLB
        ROLA
        ADDB    ,S+
        STD     CNST
        JMP     NEW
OUT     LDY     #$FFFF    Tempo apres CR
        LEAY    -1,Y
        CMPY    #00
        BNE     *-6
        CALL    $0A       Etat d'attente CR
* Permet de figer l'image en cours
        TSTB
        BEQ     OUT
        CMPB    #13T
        BNE     OUT
        PULS    A,B,X,Y,U,DP
        STOP
```

```
CALCUL PSHS    A,B,Y
       TFR     Y,D       B = Ligne
       LDA     #40
       MUL
       ADDD    #$0000
       EXG     D,X       X=Ligne*40+$0000
       LSRA
       PORB
       LSRB
       LSRB              D=Colonne/8
       LEAX    D,X       +Ligne*40
       PULS    A,B,Y,PC
       END     START
```

CONSEILS BIBLIOGRAPHIQUES

BUI Minh Duc
Programmation en assembleur 6809
Editions EYROLLES, Paris, 1983

DARDANNE Claude
Le microprocesseur 6809 - ses périphériques et le processeur graphique 9365-66
Éditions EYROLLES, Paris, 1984

Dans la même collection

Initiation au BASIC TO7/TO7-70 — Christine et François-Marie Blondel

Le BASIC D.O.S. du TO7/TO7.70 et du MO5 — Christine et François-Marie Blondel

Un ordinateur à la maison — Jean Delcourt

Un ordinateur en fête — Serge Pouts-Lajus

Un ordinateur et des jeux — Jean-Pascal Duclos

Guide pratique de l'ordinateur personnel d'IBM — Dalloz/Emery Porte-faix/Boisgontier/Salzman

LOGO, des ailes pour l'esprit — Horacio C. Reggini

Premiers pas avec le ZX-SPECTRUM — Ian Stewart/Robin Jones

Le langage machine de ZX-SPECTRUM — Ian Stewart/Robin Jones

Plus loin avec le ZX-SPECTRUM — Ian Stewart/Robin Jones

Jeux vidéo, jeux de demain — Georges-Marie Becherraz/Alain Graber

Guide partique de l'Oric-Atmos — Michel Bussac/Robert Lagoutte

Des programmes pour votre Oric-Atmos — Michel Piot

Premiers pas avec le Commodore 64 — Ian Stewart/Robin Jones

Initiation à LOGO — Doris Avram/Michèle Weidenfeld

LOGO, Manuel de référence — Doris Avram/Tristan Savatier/Michèle Weidenfeld et l'équipe de S.O.L.I

Guide du MO5 — André Deledicq

Faites vos jeux en assembleur sur TO7 et TO7-70 — Michel Oury

Manuel de l'Assembleur du 6809 et du TO7 et TO7-70 — Michel Weissgerber

Manuel de l'assembleur du 6809 du MO5 — Michel Weissgerber

Initiation au FORTH — S.E.F.I.

Guide pratique du vidéotex et du minitel — Jean-Pierre Saboureau/Geneviève Bouché

Guide pratique de l'enseignement assisté par ordinateur — Jean-Michel Lefèvre

La face cachée du TO7/TO7-70 — Jean-Baptiste Touchard

Manuel technique du TO7/TO7-70 — Michel Oury/Jean-Baptiste Touchard

Manuel technique du MO5 — Jean-Baptiste Touchard

Achevé d'imprimer en Novembre 1984
par l'Imprimerie Pollina, 85400 Luçon - N° 6609
Dépôt légal : Novembre 1984
Imprimé en France